MINIMIZING DNF FORMULAS AND AC^0 CIRCUITS GIVEN A TRUTH TABLE [*]

ERIC ALLENDER[†], LISA HELLERSTEIN[‡], PAUL MCCABE [§], TONIANN PITASSI [¶], AND MICHAEL SAKS[‖]

Abstract. For circuit classes R, the fundamental computational problem *Min-R* asks for the minimum R-size of a Boolean function presented as a truth table. Prominent examples of this problem include *Min-DNF*, which asks whether a given Boolean function presented as a truth table has a k-term DNF, and *Min-Circuit* (also called MCSP), which asks whether a Boolean function presented as a truth table has a size k Boolean circuit. We present a new reduction proving that *Min-DNF* is NP-complete. It is significantly simpler than the known reduction of Masek [31], which is from *Circuit-SAT*. We then give a more complex reduction, yielding the result that *Min-DNF* cannot be approximated to within a factor smaller than $(\log N)^{\gamma}$, for some constant $\gamma > 0$, assuming that NP is not contained in quasipolynomial time. The standard greedy algorithm for *Set Cover* is often used in practice to approximate *Min-DNF*. The question of whether *Min-DNF* can be approximated to within a factor of $o(\log N)$ remains open, but we construct an instance of *Min-DNF* on which the solution produced by the greedy algorithm is $\Omega(\log N)$ larger than optimal. Finally, we extend known hardness results for *Min-TC$_d^0$* to obtain new hardness results for *Min-AC$_d^0$*, under cryptographic assumptions.

Key words. Machine learning theory, complexity theory, approximation algorithms, truth table minimization.

AMS subject classifications. 68Q17,68Q32,03D15

1. Introduction. A fundamental computational problem is to determine the minimum size of a Boolean function in some representation, given a truth table for the function. Two prominent examples are *Min-DNF*, which asks whether a Boolean function presented as a truth table has a k-term DNF; and *Min-Circuit* (also called MCSP, for Minimum Circuit Size Problem), which asks whether a Boolean function presented as a truth table has a size k Boolean circuit. By varying the representation class, we can obtain a hierarchy of problems between *Min-DNF* and *Min-Circuit*, including such problems as *Min-AC^0*, *Min-TC^0*, and *Min-NC^1*.

The main focus of this paper is the *Min-DNF* problem. *Min-DNF* is the decision version of finding the smallest DNF formula consistent with a truth table, where the size of a DNF formula is considered to be the number of terms in it. This is a classic problem in computer science and circuit design. Heuristic approaches to solving this problem range from the Karnaugh maps of the 1960's to state-of-the-art software packages (cf. [14]).

Masek proved *Min-DNF* to be NP-complete in the 1970's [31]. This result was cited by Garey and Johnson [19] and is widely known, but Masek never published his proof. More recently, Czort presented a modernized, more readable version of Masek's proof [15] (see also [43]). Masek's proof is by direct reduction from *Circuit-SAT*, using gadget constructions, and even in Czort's version it is long and involved. We present a new, simple NP-completeness proof for *Min-DNF* by reduction from *3-Partite Set Cover* (or, more particularly, from *3D-*

Matching).

It is well-known that *Min-DNF* can be viewed as a special case of *Set Cover*, and that the greedy *Set Cover* algorithm can be applied to *Min-DNF* to produce a DNF with $O(\log N)$ times as many terms as the optimal, where N is the size (number of entries) of the input truth table. This prompts the question of whether a better approximation factor can be achieved. Czort considered this question, but showed only that unless $P = NP$, the size of the smallest DNF cannot be approximated to within an *additive* constant k [15]. We also give a more complicated reduction (again from a restricted version of *Set Cover*) that allows us to prove the following inapproximability result for *Min-DNF*: If NP is not contained in quasipolynomial time, then *Min-DNF* cannot be approximated to within a factor smaller than $(\log N)^\gamma$ for some constant $\gamma > 0$, where N is the size of the input truth table.

There is a gap between our $\Omega((\log N)^\gamma)$ inapproximability lower bound for *Min-DNF*, and the $O(\log N)$ upper bound of the greedy *Set Cover* algorithm. Closing this gap remains an open question. We do, however, construct an instance of *Min-DNF* for which the greedy *Set Cover* algorithm produces a DNF formula that has $\Omega(\log N)$ times as many terms as the optimal. The greedy *Set Cover* algorithm is commonly used as a heuristic for solving *Min-DNF* in practice. We also prove an $\Omega(\sqrt{\log N})$ inapproximability lower bound for *Min-DNF* under the additional assumption that a restriction of *Set Cover* is $\Omega(\log n)$-hard to approximate.

Although the general *Min-DNF* problem is NP-hard, for $k = O(\sqrt{\log N})$ it is tractable [20]. Using a simple padding argument, we show hardness results for *Min-DNF* where $k = \omega(\log N)$. The question of whether *Min-DNF* is tractable for $k = \log N$ remains open. This question was posed in [20]; a negative result would imply that $\log n$-term DNF cannot be learned with membership and proper equivalence queries.

In addition to our results for *Min-DNF*, we also prove a result for *Min-AC$_d^0$* for all sufficiently large d. Under cryptographic assumptions, it is known that *Min-Circuit*, *Min-NC^1* and *Min-TC$_d^0$* are not polynomial-time solvable [4]. (Nothing is stated explicitly in [4] regarding *Min-TC$_d^0$*, but it is implicit.) We extend the hardness results for *Min-TC$_d^0$* to obtain new hardness results for *Min-AC$_d^0$*, under cryptographic assumptions. This still leaves open the interesting question of whether *Min-Circuit* (or the other problems) are NP-complete. Kabanets and Cai [24] give evidence that such a reduction will not be straightforward.

The organization of this paper is as follows. In Section 2 we define the relevant minimization problems and present necessary background. In Section 3 we present our new proof that *Min-DNF* is NP-hard. In Section 4 we present our hardness results for approximating *Min-DNF*. In Section 5 we give our construction of the instance of *Min-DNF* on which the greedy *Set Cover* algorithm produces an $\Omega(\log N)$ factor approximation. Section 6 concerns the fixed parameter versions of *Min-DNF*. Our hardness results for *Min-AC$_d^0$* appear in Section 8. Conclusions are in Section 9.

A preliminary version of this paper appeared in [3]. Feldman independently proved an $\Omega((\log N)^\delta)$ factor inapproximability result for *Min-DNF* [18] using related techniques. Feldman's result is based on the assumption $P \neq NP$, rather than on the assumption that NP is not contained in quasipolynomial time. Feldman also proved new results on proper learning of DNF, which are discussed in Section 7.

2. Preliminaries. We begin with a few definitions. The set $\{1, \ldots, n\}$ is denoted by $[n]$. We use the bitwise ordering on vectors: for $v, w \in \{0, 1\}^n$, we write $v \leq w$ if $v_i \leq w_i$ for all $i \in [n]$. Let $V_n = \{x_1, \ldots, x_n\}$. A *prime implicant* T of a function $f(x_1, \ldots, x_n)$ is a conjunction of literals over the variables V_n such that $T = 1 \Rightarrow f = 1$, and removing any literal from T violates this property. (In the literature, prime implicants are sometimes called *minterms*). A *DNF formula* over the variables V_n is a formula $\phi = T_1 \vee T_2 \vee \ldots T_k$ for some k, where $T_1, \ldots, T_k$ are each conjunctions of literals over V_n. Each T_i in ϕ is a *term* of ϕ. Every Boolean

function f can be expressed by a DNF formula in which every term is a prime implicant of f. The *size* of a DNF formula is the number of terms in it; for a Boolean function or partial function f, *dnf-size(f)* denotes the size of the smallest DNF formula consistent with f. The class of Boolean circuits AC_d^0 consists of all depth-d circuits of AND and OR gates with arbitrary fan-in.

The classic *Set Cover* optimization problem is, given input $(\mathscr{S}, \mathscr{U})$, where $\mathscr{U}$ is a finite universe, and $\mathscr{S}$ is a collection of subsets of $\mathscr{U}$, find a smallest subcollection $\mathscr{C} \subseteq \mathscr{S}$, such that that the union of the sets in $\mathscr{C}$ equals $\mathscr{U}$. It is NP-hard to approximate *Set Cover* to within a factor smaller than $c \log n$, where c is a constant and n is the size of the input (cf. [7]). On the other hand, there is a simple greedy algorithm that achieves an $O(\log n)$ approximation for *Set Cover* [22, 29, 13].

For r a positive integer, the *r-Uniform Set Cover* problem is as follows: on input $(n, k, \mathscr{S})$ where n and k are positive integers and $\mathscr{S}$ is a set of subsets of $[n]$, each subset having size r, determine whether there is a subcollection $\mathscr{C} \subseteq \mathscr{S}$ of size at most k whose union is $[n]$. The *r-Partite Set Cover* problem is a restriction: on input $(n, k, \Pi, \mathscr{S})$ where n and k are positive integers, Π is a partition of $[n]$ into r sets, and $\mathscr{S}$ is a collection of subsets of $[n]$, where every subset contains exactly one element from each of the sets of Π, determine whether there is a subcollection $\mathscr{C} \subseteq \mathscr{S}$ of size at most k whose union is $[n]$. The *3D-Matching* problem is the NP-complete restriction of *3-Partite Set Cover* where $k = n/3$ (cf. [19]).

We consider a general family of computational problems of the form *Min-R(S)* where the input is a Boolean function with input representation from *S*, and the output should be a minimum representation of the function from *R*. For example, *Min-DNF(tt)* is the problem of determining a smallest DNF representation of a Boolean function f on n variables, if f is presented as a truth table of size $N = 2^n$. Our default input representation will be the truth table representation and when we write *Min-R*, rather than *Min-R(S)*, we will assume the default input representation.

We focus primarily on DNF minimization. We consider the following four variations:

Min-DNF(A): The input is a total Boolean function, specified by explicitly listing all 1's of the function. That is, $A \subseteq \{0,1\}^n$ is the input, and we look for a minimum DNF that realizes the total function f_A, where $f_A(a) = 1$ for $a \in A$, and $f_A(b) = 0$ for $b \in \{0,1\}^n - A$.

Min-DNF: In the full-truth table version, the input is the entire truth table of $f : \{0,1\}^n \to \{0,1\}$, and we look for a minimum DNF that realizes the function f.

Min-DNF(A,B): The input is a partial Boolean function, specified by listing the 1's and 0's of the function, and we look for a minimum DNF that is consistent with the input. That is, $A, B \subseteq \{0,1\}^n$ is the input, and we look for a minimum DNF that realizes a function $f : \{0,1\}^n \to \{0,1\}$, where $f(a) = 1$ for $a \in A$ and $f(b) = 0$ for $b \in B$.

Min-DNF(*): The input is a partial Boolean function, specified by the entire truth table of $f : \{0,1\}^n \to \{0,1,*\}$, where $f(a) = *$ means that the value of f is not defined on a. We look for a minimum DNF that realizes a function $f' : \{0,1\}^n \to \{0,1\}$, where $f'(a) = 1$ for $a \in f^{-1}(1)$ and $f'(b) = 0$ for $b \in f^{-1}(0)$. Note that as in the (A, B) version, the input here also specifies a partial function, but now the partial function is specified by a 2^n sized input, regardless of the size of the domain of the partial function.

The decision versions of the above problems ask, given a function f and a natural number k, whether or not there is a DNF formula realizing f that has at most k terms. All decision versions are easily seen to lie in NP. It is also easy to see that *Min-DNF* is a special case of *Min-DNF(*)* and therefore reduces to *Min-DNF(*)*, and *Min-DNF(*)* reduces to *Min-DNF(A,B)*. Also *Min-DNF* reduces to *Min-DNF(A)*. Thus NP-hardness of *Min-DNF* implies NP-hardness

of all other versions. The first three of the above problems are covered by Czort [15] in an excellent survey of previous related work. There is a hodgepodge of interesting but incomparable hardness results that are known for versions of DNF minimization, dating back to the 1960's. The simplest of these is the NP-hardness of the (A, B) version due to Pitt and Valiant [35]. As shown by Czort, there is also a clean NP-hardness proof of the A version that follows from a reduction of Gimpel. Masek [31] proved the NP-completeness of *Min-DNF*. In terms of inapproximability, Pitt and Valiant's proof of the (A, B) hardness result preserves solution values and thus shows the NP-hardness of achieving a factor n^ε approximation. Neither of the other two NP-hardness proofs (for the A version or for *Min-DNF*) give much in the way of inapproximability results.

A starting point for this paper is the well-known observation that *Min-DNF* easily reduces to *Set Cover*, and in fact can be viewed as a special case of *Set Cover*. Given the truth table of a Boolean function f over n variables, all prime implicants of f can be generated in time $2^{O(n)}$. Each prime implicant can then be viewed as a subset of $\{0, 1\}^n$ (corresponding to those inputs that satisfy the prime implicant). Thus given all the prime implicants, finding a smallest DNF is equivalent to finding a smallest cover for these prime implicant sets. Applying the standard greedy algorithm for *Set Cover*, it follows that *Min-DNF* can be approximated to within a factor of $O(\log N)$, where N is the size (number of entries) of the truth table.

For a partial Boolean function f, the prime implicants of f are the prime implicants of the total function f' that satisfies $f'(\vec{x}) = 1$ iff $f(\vec{x}) = 1 \vee f(\vec{x}) = *$. Every partial function f has a smallest consistent DNF whose terms are prime implicants of f. The greedy *Set Cover* algorithm can also be used to approximate *Min-DNF(*)* in the same way that it is applied to *Min-DNF*, except that it chooses sets that cover the maximum number of 1's of the input function (i.e. it ignores $*$'s when greedily choosing sets).

The pseudocode for applying the greedy *Set Cover* algorithm to *Min-DNF* and *Min-DNF(*)* is shown below. The input is the full truth table of a Boolean function or partial function f.

1: $\mathscr{T} := \{ T \mid T \text{ is a prime implicant of } f \}$
2: $\varphi := \bot$
3: **while** φ does not cover all 1's of f **do**
4:      let $T \in \mathscr{T}$ cover the most uncovered 1's of f
5:      $\varphi := \varphi \vee T$
6: **end while**
7: return φ

FIG. 2.1. *Greedy Min-DNF and Min-DNF(*) algorithm*

3. Simple proof that *Min-DNF* is NP-complete. Our new proof that *Min-DNF* is NP-complete is a modification of the reduction of Gimpel mentioned above, which was was used by Czort to prove the NP-completeness of the A version of DNF minimization [15].

We start by briefly describing Gimpel's reduction. It can be viewed as consisting of two phases. In the first phase, an instance $(\mathscr{S}, \mathscr{U})$ of *Set Cover* over the ground set $\mathscr{U} = [n]$ is mapped to a partial function f, as follows. First, both the sets as well as the ground elements are mapped to truth assignments in $\{0, 1\}^n$, such that a set covers a ground element in $[n]$ if and only if the assignment corresponding to the ground element is less than the assignment corresponding to the set (where comparison of assignments is with respect to the bitwise ordering of the vectors). Each ground element $i \in [n]$ is mapped to the assignment that is all zero except for bit i, which is 1. Each set is mapped to the assignment corresponding to the

characteristic function of the set. The 1's of f are those assignments corresponding to ground elements; the $*$'s of f are those assignments α such that $\alpha \leq \beta$ for some β corresponding to a set; and the remaining truth assignments are zeroes of f. It can be shown that the size of the minimum DNF consistent with the partial function f is equal to the minimum size of the cover for the input instance of *Set Cover*.

In the second phase of Gimpel's reduction, the partial function f is mapped to a total function, g. We give the details of g below in Section 3.2. The truth table size of f and g are exponential in the size of the *Set Cover* instance from the first phase. Thus Gimpel's reduction does not give a hardness result for *Min-DNF*. As Czort notes, it does, however, give a hardness result for *Min-DNF(A)*, provided that we begin the reduction not from the general *Set Cover* problem, but from *3-Uniform Set Cover* .

Our reduction proving that *Min-DNF* is NP-complete also has two phases. The first phase is similar to that of Gimpel. The main difference is that we need a much more compact mapping from the sets and ground elements of the *Set Cover* instance onto truth assignments, to ensure that the size of the truth table for the resulting function is only polynomial in the size of the input *Set Cover* instance. To do such a compact mapping in a simple way, we reduce from *3-Partite Set Cover*, rather than from *3-Uniform Set Cover*. The second phase of our reduction is essentially identical to Gimpel's.

3.1. Reducing *3-Partite Set Cover* **to** *Min-DNF(*)*. In the first phase of our reduction, we reduce *3-Partite Set Cover* to *Min-DNF(*)*. We note that our reduction from *3-Partite Set Cover* would also work from *3D-Matching*. We use the following lemma, which is implicit in Gimpel's reduction:

LEMMA 3.1. *Let $\mathscr{S}$ be a set of subsets of $[n]$. Let $t > 0$ and let $V = \{v^i : i \in [n]\}$ and $W = \{w^A : A \in \mathscr{S}\}$ be sets of vectors from $\{0,1\}^t$ satisfying*
        (*) For all $A \in \mathscr{S}$ and $i \in [n]$, $i \in A$ iff $v^i \leq w^A$
Let $R = \{x \in \{0,1\}^t \mid x \notin V$ and for some $w \in W$, $x \leq w\}$. Let f be a partial function with domain $\{0,1\}^t$ such that $f(x) = 1$ if $x \in V$, $f(x) = *$ if $x \in R$, and $f(x) = 0$ otherwise. Then $\mathscr{S}$ has a cover of size m if and only if there is an m-term DNF consistent with f.

Proof. For $u \in \{0,1\}^t$, let $D(u) = \{w : w \leq u\}$ and let $\tau(u)$ denote the DNF term $\bigwedge_{i:u_i=0} \neg x_i$. Note that $D(u)$ is exactly the set of satisfying assignments of $\tau(u)$. For a set U of vectors $D(U) = \bigcup_{u \in U} D(u)$. By (*), we have that $V \subseteq D(W)$. Also, $f(x) = *$ iff $x \in D(W) - V$.

Given a cover $\mathscr{C} \subseteq \mathscr{S}$ of size m, the m-term DNF whose terms are $\{\tau(w^C) \mid C \in \mathscr{C}\}$ is easily seen to be consistent with f. Conversely, suppose ϕ is an m-term DNF consistent with f. For each term $\tau \in \phi$, let $u(\tau)$ be the maximal vector satisfying τ. Since ϕ is consistent with f, we have that $u(\tau) \in D(W)$, so there must be a set $S(\tau) \in \mathscr{S}$ for which $u(\tau) \leq w^{S(\tau)}$. We claim that $\{S(\tau) : \tau \in \phi\}$ is a cover of $\mathscr{S}$. Let $j \in [n]$. We must show that j is covered. The consistency of ϕ implies that v^j is satisfied by some term $\tau_j \in \phi$. This implies $v^j \leq u(\tau_j)$. Thus $v^j \leq w^{S(\tau_j)}$, which by (*) implies $j \in S(\tau_j)$. $\square$

The reduction from *3-Partite Set Cover* to *Min-DNF(*)* is given in the following lemma.

LEMMA 3.2. *There is an algorithm that takes as input an instance $(n, k, \Pi, \mathscr{S})$ of 3-Partite Set Cover and outputs an instance of Min-DNF(*). The instance of Min-DNF(*) defines a partial function f on $O(\log n)$ variables, such that the size of the smallest DNF consistent with f is equal to the size of the smallest cover for the input 3-Partite Set Cover instance. The algorithm runs in time polynomial in n.*

Proof. Given an input instance $(n, k, \Pi, \mathscr{S})$ of *3-Partite Set Cover*, the algorithm produces an indexed set of vectors $V = \{v^i : i \in [n]\}$ and $W = \{w^A : A \in \mathscr{S}\}$ all of the same

(small) length t satisfying the condition (*) of Lemma 3.1. We will specify V and then define W according to the rule that for $A \in \mathscr{S}$, w^A is the bitwise OR of $\{v^i : i \in A\}$. This guarantees the forward implication of condition (*) for any choice of V; it is the backward implication that requires some care in choosing V.

Let q be the smallest integer such that $\binom{q}{q/2} \geq n$. Thus $q = O(\log n)$. Assign to each $i \in [n]$ a unique q-bit Boolean vector $b(i)$ containing exactly $q/2$ 1's. For $i \in [n]$, write $\Pi(i)$ for the index of the block of Π that contains i. Let $t = 3q$. We will consider the t-bit vectors in V and W as being divided into 3 blocks of size q. For $i \in [n]$, let v^i be equal to 0 on all blocks but block $\Pi(i)$; on block $\Pi(i)$ it is $b(i)$. To see that the backward implication of (*) holds, let $A \in \mathscr{S}$ and $i \in [n]$ and assume that $v^i \leq w^A$. Then A contains one element i' with $\Pi(i') = \Pi(i)$ and so we must have $b(i) \leq b(i')$, which implies $i = i'$.

V and W can be generated in time $n^{O(1)}$. The partial function f will have domain $\{0,1\}^t$. The lemma then follows immediately from Lemma 3.1. □

3.2. Reducing *Min-DNF(*)* **to** *Min-DNF*. As mentioned above, the second phase of our reduction is taken from Gimpel. We describe the phase here, and will build on it later in order to prove inapproximability results. The second phase of Gimpel's reduction maps a partial function f to a total function g. The variables underlying g are V (the variables of f) plus two additional variables, y_1 and y_2. The total function g is defined as follows:

$$g(\vec{x}\, y_1 y_2) = \begin{cases} 1, & \text{if } f(\vec{x}) = 1 \text{ and } y_1 = y_2 = 1 \\ 1, & \text{if } f(\vec{x}) = * \text{ and } y_1 = y_2 = 1 \\ 1, & \text{if } f(\vec{x}) = *, y_1 = p(\vec{x}), \text{ and } y_2 = \neg p(\vec{x}) \\ 0, & \text{otherwise} \end{cases}$$

where $p(\vec{x}) = 0$ if the parity of $\vec{x}$ is even, and $p(\vec{x}) = 1$ if the parity of $\vec{x}$ is odd. Let $s = |f^{-1}(*)|$. The following lemma is implicit in Gimpel's reduction (cf. [15]).

LEMMA 3.3. *dnf-size*$(g) = $ *dnf-size*$(f) + s$.

Proof. The idea behind the proof is as follows. The key observation is that every DNF for g requires s distinct terms to cover the inputs of the third type in the definition of g above; these terms can simultaneously cover all inputs of the second type, but not those of the first type. The remaining terms of the DNF must therefore cover the terms of the first type; and may optionally cover the terms of the second type; they thus constitute a solution to the *Min-DNF(*)* problem for f. It follows that *dnf-size*$(g) = $ *dnf-size*$(f) + s$. We now prove this formally.

We first show that *dnf-size*$(g) \leq $ *dnf-size*$(f) + s$. Suppose φ is a minimum-size DNF consistent with f. Define a DNF ψ with terms of two types: first, for every input $\vec{x} \in f^{-1}(*)$, ψ contains the term $\left(\bigwedge_{i:\vec{x}_i=1} x_i \right) \wedge \left(\bigwedge_{i:\vec{x}_i=0} \neg x_i \right) \wedge y_{2-p(\vec{x})}$. These terms cover all inputs of the second and third types in the definition of g. Second, for every term T of φ, ψ contains the term $T \wedge y_1 \wedge y_2$. These terms cover all inputs of the first type in the definition of g.

Finally, suppose that $\vec{x} y_1 y_2$ satisfies ψ. Then one of the following three conditions holds: (1) $\vec{x} \in f^{-1}(*)$, $y_1 = p(\vec{x})$, and $y_2 = \neg p(\vec{x})$, (2) $\vec{x} \in f^{-1}(*)$, and $y_1 = y_2 = 1$, (3) $\vec{x}$ satisfies φ (and thus $\vec{x} \in f^{-1}(1) \cup f^{-1}(*)$) and $y_1 = y_2 = 1$. In all three cases we have $g(\vec{x} y_1 y_2) = 1$, and thus ψ is consistent with g. The number of terms in ψ is $|f^{-1}(*)| + |\varphi| = $ *dnf-size*$(f) + s$.

We now show that *dnf-size*$(g) \geq $ *dnf-size*$(f) + s$. Suppose that ψ is a smallest DNF for g. We assume without loss of generality that each term of ψ is a prime implicant of g. We begin by proving that, for every $\vec{x} \in f^{-1}(*)$, ψ contains the term $t(\vec{x}) \wedge y_{2-p(\vec{x})}$, where

$t(\vec{x}) = \left(\bigwedge_{i:\vec{x}_i=1} x_i\right) \wedge \left(\bigwedge_{i:\vec{x}_i=0} \neg x_i\right)$. The proof is as follows. Let $\vec{x} \in f^{-1}(*)$, and suppose that the parity of $\vec{x}$ is odd: the case of even parity is symmetric. Let T be a term of ψ that is satisfied by $\vec{x}10$ (where 1 and 0 are the values of y_1 and y_2 respectively). If, for some index i, T does not contain the variable x_i, let $\vec{x}'$ be obtained by flipping the i-th bit of $\vec{x}$. Then $\vec{x}'10$ falsifies g (since $\vec{x}'$ has even parity), but satisfies T, contradicting the assumption that ψ is consistent with g. Thus T contains each of the variables $x_1, \ldots, x_n$. In addition, T contains the variable y_1, as otherwise $\vec{x}00$ would satisfy T. Finally, since T is a prime implicant of g, we have that $T = t(\vec{x})y_1$.

We now prove that there exists a subformula $\hat{\psi}$ of ψ and a DNF ψ' over the $\vec{x}$ variables that is consistent with f, such that $\hat{\psi} = \bigvee_{T \in \psi'}(T \wedge y_1 \wedge y_2)$. Let $\hat{\psi}$ be the subformula of ψ consisting of those terms that are satisfied by $\vec{x}11$ for some $\vec{x} \in f^{-1}(1)$. Each term of $\hat{\psi}$ contains $y_1 \wedge y_2$, since flipping y_1 or y_2 produces an input that falsifies g. It follows that $\hat{\psi} = \bigvee_{T \in \psi'}(T \wedge y_1 \wedge y_2)$ where ψ' is a DNF. It remains to show that ψ' is consistent with f. For every $\vec{x} \in f^{-1}(1)$, there is a term of ψ satisfied by $\vec{x}11$, and thus there is a corresponding term of ψ' satisfied by $\vec{x}$. On the other hand, every $\vec{x} \in f^{-1}(0)$ must falsify ψ', as otherwise $\vec{x}11$ would satisfy ψ.

It follows from the above that ψ consists of the terms $t(\vec{x}) \wedge y_{2-p(\vec{x})}$ for each $\vec{x} \in f^{-1}(*)$, together with the subformula $\hat{\psi}$. These components are pairwise disjoint. Since ψ' is consistent with f, $\hat{\psi}$ contains at least *dnf-size*(f) terms, and thus the size of ψ is at least *dnf-size*$(f) + s$. $\square$

It follows that there is a polynomial-time reduction from *Min-DNF(*)* to *Min-DNF*. Combining this with the previous reduction from *3-Partite Set Cover* to *Min-DNF(*)*, it follows that *Min-DNF* is NP-complete.

4. On the Approximability of *Min-DNF*. Although the two-phase reduction above proves the NP-completeness of *Min-DNF*, it does not give us inapproximability results. There are two problems. First, the reduction begins with an instance of *3-Partite Set Cover*, a problem that can be approximated in polynomial time to within a factor of $\ln 3$ [22, 29] (since the size of the largest subset is 3); to obtain inapproximability results we need to reduce from a problem that is difficult to approximate. Also, the second phase of the reduction, from *Min-DNF(*)* to *Min-DNF*, is not approximation preserving.

We replace the first phase of the reduction with a reduction that exploits properties of the *Set Cover* instance obtained by the PCP-based inapproximability results of Lund/Yannakakis and Feige [30, 17]. We then modify the second phase to make it approximation preserving. The final two-phase reduction gives an inapproximability factor of $\Omega((\log N)^\gamma)$ assuming that NP is not contained in quasipolynomial time.

In Appendix A we also present a modified version of the first phase of the reduction, which reduces from *r-Uniform Set Cover* rather than from *3-Partite Set Cover*. This allows us to obtain an inapproximability result for *Min-DNF* by applying known inapproximability results for *r-Uniform Set Cover*. However, the result we obtain for *Min-DNF* is weak (inapproximability to within a factor of $\Omega(\log \log N)$). Nevertheless, the reduction itself may be of independent interest, since it requires a different technique to reduce from *r-Uniform Set Cover* rather than from *r-Partite Set Cover*.

4.1. New reduction to *Min-DNF(*)*. In this section we present a reduction that follows the PCP-based inapproximability results for *Set Cover* [30, 17]. We will closely follow the Lund/Yannakakis reduction, as presented by Khot [26].

An instance of *Label Cover* is denoted by $\mathscr{L} = (G, L_1, L_2, \Pi)$ where $G = (V, W, E)$ is a regular bipartite graph, L_1 and L_2 are sets of labels, and $\Pi = \{\pi_{vw}\}_{(v,w) \in E}$ denotes the

constraints on each edge. For every edge $(v, w) \in E$ we have a map $\pi_{vw} : L_1 \to L_2$. A labelling $l : V \to L_1, W \to L_2$ satisfies the constraint on an edge (v, w) if $\pi_{vw}(l(v)) = l(w)$. Given an instance $\mathcal{L}$, the output should be a labelling that satisfies the maximum fraction, $\text{OPT}(\mathcal{L})$, of edge constraints.

THEOREM 4.1. **[30, 26]** *There is a constant $c < 1$ such that it is NP-hard to solve the following gap version of Label Cover. The input is an instance $\mathcal{L} = (G = (V, W, E), [7], [2], \{\pi_{vw}\}_{(v,w) \in E})$ of Label Cover. The instance should be accepted if $\text{OPT}(\mathcal{L}) = 1$, and the instance should be rejected if $\text{OPT}(\mathcal{L})$ is at most c.*

Note that the reduction is from *Max3SAT(5)*(the problem of maximizing the number of satisfied clauses in a 3CNF formula where each variable occurs in exactly five clauses). The vertices in V correspond to the m clauses, and the vertices in W correspond to the n variables. Using Raz's parallel repetition theorem [36], we can amplify the gap, obtaining, for any positive integer k, an instance $\mathcal{L}' = (G' = (V', W', E'), [7^k], [2^k], \{\pi_{v'w'}\}_{(v',w') \in E'})$, where $|V'| = |V|^k$ and $|W'| = |W|^k$, such that $\text{OPT}(\mathcal{L}) = 1$ implies $\text{OPT}(\mathcal{L}') = 1$, and $\text{OPT}(\mathcal{L}) \leq c$ implies $\text{OPT}(\mathcal{L}') \leq 2^{-\gamma k}$, where $\gamma > 0$ is an absolute constant. Note that the sizes of both V' and W' are $n^{O(k)}$, where n is the number of variables in the *Max3SAT(5)* instance.

DEFINITION 4.2. *A partition system $\mathscr{P}(m, h, t)$ consists of t partitions $(A_1, \overline{A_1}), \ldots (A_t, \overline{A_t})$ of $[m]$, with the property that no collection of h sets, with at most one set from each partition, covers all of $[m]$.*

LEMMA 4.3. **[30]** *For every h and t, there is an efficiently constructible partition system $\mathscr{P}(m, h, t)$ with $m = O(2^h h \log t)$.*

We now review the reduction from the *Label Cover* instance $\mathcal{L}'$ to a *Set Cover* instance $(\mathscr{S}, \mathscr{U})$. First, the universe $\mathscr{U}$ is as follows. Let $t = 2^k$, let h be a parameter to be determined later, and let $m = m(t, h) = O(2^h h \log t)$ be the parameter specified by Lemma 4.3. For each edge $e \in E'$ we associate a subuniverse $\mathscr{U}_e = \{(e, i) \mid i \in [m]\}$. The entire universe $\mathscr{U}$ is the disjoint union of these $|E'|$ subuniverses. Associated with each edge e is a partition system $\mathscr{P}(m, h, t)$ over $\mathscr{U}_e$, with one partition associated with each of the possible labels in L_2. Thus each label $b \in [t]$ corresponds to a partition $(A_b^e, \overline{A_b^e})$ of $\mathscr{U}_e$. The size of the entire universe is $n^{O(k)} 2^{O(h)}$. The set system $\mathscr{S}$ is the union of two collections of sets: $S(v, a)$, for each vertex $v \in V'$ and each label $a \in [7^k]$; and $S(w, b)$, for each $w \in W'$ and each label $b \in [2^k]$. In particular,

$$S(v, a) = \bigcup_{w : (v,w) \in E'} A_{\pi_{vw}(a)}^{(v,w)} \qquad\qquad S(w, b) = \bigcup_{v : (v,w) \in E'} \overline{A_b^{(v,w)}}.$$

The following lemma is implicit in [30, 26].

LEMMA 4.4. **[30, 26]** *If $\text{OPT}(\mathcal{L}') = 1$ then $(\mathscr{S}, \mathscr{U})$ has a cover of size $|V'| + |W'|$. If $\text{OPT}(\mathcal{L}') \leq 1/(2h^2)$ then every cover of $(\mathscr{S}, \mathscr{U})$ has size at least $h(|V'| + |W'|)/16$.*

Choosing $h \leq 2^{\gamma k/2 - 1/2}$, we obtain a gap of $h/16$ for the *Set Cover* instance from the $2^{\gamma k}$ gap of the *Label Cover* instance. For $k = O(\log \log n)$ sufficiently large, we have $|\mathscr{U}| = 2^{O(h)}$, and thus the gap is $\Omega(\log |\mathscr{U}|)$. The size of the *Set Cover* instance is quasipolynomial in n. Thus a polynomial-time, $(h/16)$-approximation algorithm for *Set Cover* could distinguish between the cases $\text{OPT}(\mathcal{L}') = 1$ and $\text{OPT}(\mathcal{L}') \leq 2^{-\gamma k}$ in time $2^{\text{polylog}(n)}$, implying that NP is contained in $DTIME(2^{\text{polylog}(n)})$.

We now show how to reduce instances of *Set Cover* of the above form to *Min-DNF(*)* instances. By the observations in Section 3.1 it suffices to define three sets of vectors,

$\{u_{e,i} \mid (e,i) \in \mathcal{U}\}$, $\{t_{v,a} \mid v \in V', a \in L_1\}$, and $\{t_{w,b} \mid w \in W', b \in L_2\}$ such that the following conditions hold: (1) $u_{e,i} \leq t_{v,a}$ iff $(e,i) \in S(v,a)$, for all $(e,i) \in \mathcal{U}$, $v \in V$ and $a \in L_1$; and (2) $u_{e,i} \leq t_{w,b}$ iff $(e,i) \in S(w,b)$, for all $(e,i) \in \mathcal{U}$, $w \in W$, and $b \in L_2$, Let $r \in O(\log|V'|)$ be such that $\binom{r}{r/2} \geq \max(|V'|,|W'|)$. Our function will have variables $\{x_1,\ldots,x_r\} \cup \{x'_1,\ldots,x'_r\} \cup \{y_a \mid a \in L_1\} \cup \{y'_b \mid b \in L_2\}$. Thus the number of variables is $O(\log|V'| + |L_1| + |L_2|) = O(k\log n + 7^k)$.

We assign to each $v \in V'$ a unique set $S_v \subseteq \{1,\ldots,r\}$ of size $r/2$; and similarly each $w \in W'$ is assigned a unique set $S_w \subseteq \{1,\ldots,r\}$ of size $r/2$. For each $v \in V'$ and $a \in L_1$, we define a Boolean vector $t_{v,a}$ as follows. The vector $t_{v,a}$ has zeroes corresponding to those variables x_i such that $i \in S_v$; and it has a zero corresponding to y_a. The remaining bits of $t_{v,a}$ are ones. We similarly define, for each $w \in W'$ and $b \in L_2$, a Boolean vector $t_{w,b}$ having zeroes corresponding to those variables x'_i such that $i \in S_w$, and a zero corresponding to y'_b, and whose remaining bits are ones.

We now describe, for each $(e,i) \in \mathcal{U}$, a Boolean vector $u_{e,i}$. Suppose that $e = (v,w)$, and let $S(v,a_1),\ldots,S(v,a_k)$ and $S(w,b_1),\ldots,S(w,b_\ell)$ be all of the sets in $\mathscr{S}$ containing (e,i). Then $u_{e,i}$ has zeroes in the positions corresponding to the following variables: (1) Variables x_i, where $i \in S_v$, (2) Variables x'_i, where $i \in S_w$, (3) Variables y_{a_i}, where $1 \leq i \leq k$, and (4) Variables y'_{b_i}, where $1 \leq i \leq \ell$. The remaining bits of $u_{e,i}$ are ones.

LEMMA 4.5. *For all $(e,i) \in \mathcal{U}$, $v \in V$, $w \in W$, $a \in L_1$, and $b \in L_2$, the following conditions hold: $u_{e,i} \leq t_{v,a}$ iff $(e,i) \in S(v,a)$ and $u_{e,i} \leq t_{w,b}$ iff $(e,i) \in S(w,b)$.*

Proof. Suppose first that $(e,i) \in S(v,a)$, where $v \in V'$ and $a \in L_1$. Then $e = (v,w)$ for some vertex $w \in W'$. The zeroes of $t_{v,a}$ are in positions corresponding to variables x_i, where $i \in S_v$, and in the position corresponding to y_a. Since $e = (v,w)$, the vector $u_{e,i}$ has zeroes in the positions corresponding to variables x_i, where $i \in S_v$, and since $(e,i) \in S(v,a)$ the vector $u_{e,i}$ also has a zero in the position corresponding to y_a. Thus $u_{e,i} \leq t_{v,a}$. The case where $(e,i) \in S(w,b)$, where $w \in W'$ and $b \in L_2$, is symmetric.

Now suppose that $(e,i) \notin S(v,a)$, where $v \in V'$ and $a \in L_1$. Suppose that $e = (v',w')$. If $v' \neq v$ then there exists an index $j \in S_v \setminus S_{v'}$; and $u_{e,i}$ has a one in the position corresponding to x_j, while $t_{v,a}$ has a zero in the same position, and thus $u_{e,i} \not\leq t_{v,a}$. So assume that $v' = v$. By definition of $t_{v,a}$, we know that $t_{v,a}$ has a zero in the position corresponding to y_a. But since $e = (v,w')$, $u_{e,i}$ has a zero in this position iff $(e,i) \in S(v,a)$; and as we have supposed that $(e,i) \notin S(v,a)$ it follows that the position in $u_{e,i}$ corresponding to y_a is set to one. Thus $u_{e,i} \not\leq t_{v,a}$. The case where $(e,i) \notin S(w,b)$, where $w \in W'$ and $b \in L_2$, is symmetric. □

By the results of Section 3.1, the vectors $u_{e,i}$, $t_{v,a}$, and $t_{w,b}$ yield an instance of *Min-DNF(*)* on $O(k\log n + 7^k)$ variables whose optimum is equal to the optimum for the instance $(\mathscr{S},\mathcal{U})$ of *Set Cover*.

THEOREM 4.6. *If $NP \not\subseteq DTIME(2^{\mathrm{polylog}(n)})$ then there exists an absolute constant $\delta > 0$ such that no polynomial time algorithm achieves an approximation ratio better than $(\log N)^\delta$ for Min-DNF(*), where N is the size of the input truth table.*

Proof. Let f be the partial function specified by our reduction. Claims 4.4 and 4.5, together with the results of Section 3.1, imply that our *Min-DNF(*)* instance has the following properties: if OPT$(\mathscr{L}') = 1$, then *dnf-size*$(f) \leq |V'| + |W'|$; and if OPT$(\mathscr{L}') \leq 2^{-\gamma k}$, then *dnf-size*$(f) \geq h(|V'| + |W'|)/16$, where $h = \Omega(2^{\gamma k/2})$. Let us take $k = \log\log n$, and thus $h = \Omega((\log n)^{\gamma/2})$. Let N be the size of the truth table for f. The number of variables of f is $\log N = O(k\log n + 7^k) = O((\log n)^{\log 7})$, and thus the gap is $h/16 = \Omega((\log N)^{\gamma/(2\log 7)})$. The

truth table has size $2^{\mathrm{polylog}\,n}$, and can be generated in time polynomial in its size. The theorem follows by taking $\delta = \gamma/(2\log 7)$. □

4.2. Approximation-preserving reduction from *Min-DNF(*)* to *Min-DNF*. We modify the reduction from Section 3.2 to make it approximation preserving. Let f be a partial Boolean function over variables $x_1, \ldots, x_n$. Let $s = |f^{-1}(*)|$. We construct a new total function g' such that $\textit{dnf-size}(g') = s \cdot \textit{dnf-size}(f) + s = s \cdot (\textit{dnf-size}(f) + 1)$. Let $t = n+1$, and let $S \subseteq \{0,1\}^t$ be a collection of s vectors, each containing an odd number of 1's. We add t new variables $z_1, \ldots, z_t$, and define

$$
g'(\vec{x}\,y_1 y_2\,\vec{z}) = \begin{cases} 1, & \text{if } f(\vec{x}) = 1,\, y_1 = y_2 = 1,\, \text{and } \vec{z} \in S \\ 1, & \text{if } f(\vec{x}) = * \text{ and } y_1 = y_2 = 1 \\ 1, & \text{if } f(\vec{x}) = *,\, y_1 = p(\vec{x}),\, \text{and } y_2 = \neg p(\vec{x}) \\ 0, & \text{otherwise} \end{cases}
$$

LEMMA 4.7. $\textit{dnf-size}(g') = s \cdot \textit{dnf-size}(f) + s$

Proof. For binary vector $\vec{w}$, we use $t(\vec{w})$ to denote the term $\left(\bigwedge_{i:w_i=1} w_i \right) \wedge \left(\bigwedge_{i:w_i=0} \neg w_i \right)$.

We first show that $\textit{dnf-size}(g') \leq s \cdot \textit{dnf-size}(f) + s$. Suppose that φ is a smallest DNF consistent with f. Define a DNF ψ with terms of the following two types. First, for every input $\vec{x} \in f^{-1}(*)$, ψ contains the term $t(\vec{x}) \wedge y_{2-p(\vec{x})}$. These terms cover all inputs of the second and third types in the definition of g'. Second, for every term T of φ and every vector $\vec{z} \in S$, ψ contains the term $T \wedge y_1 \wedge y_2 \wedge t(\vec{z})$. These terms cover all inputs of the first type in the definition of g'. Finally, suppose that $\vec{x}y_1 y_2 \vec{z}$ satisfies ψ. Then one of the following conditions holds: (1) $\vec{x} \in f^{-1}(*)$, $y_1 = p(\vec{x})$, and $y_2 = \neg p(\vec{x})$, (2) $\vec{x} \in f^{-1}(*)$, and $y_1 = y_2 = 1$, (3) $\vec{x}$ satisfies φ (and thus $\vec{x} \in f^{-1}(1) \cup f^{-1}(*)$), $y_1 = y_2 = 1$, and $\vec{z} \in S$. In all three cases we have $g'(\vec{x}y_1 y_2 \vec{z}) = 1$, and thus ψ is consistent with g'. The number of terms in ψ is $|f^{-1}(*)| + |\varphi| \cdot |S| = s \cdot \textit{dnf-size}(f) + s$.

We next show that $\textit{dnf-size}(g') \geq s \cdot \textit{dnf-size}(f) + s$. Suppose that ψ is a smallest DNF for g'. The same reasoning used in the proof of Lemma 3.3 shows that, for every $\vec{x} \in f^{-1}(*)$, ψ contains the term $t(\vec{x}) \wedge y_{2-p(\vec{x})}$. We now argue that for each $\vec{z} \in S$, there exists a subformula $\psi_{\vec{z}}$ of ψ, and a DNF $\psi'_{\vec{z}}$ over the $\vec{x}$ variables and consistent with f, such that $\psi_{\vec{z}} = \bigvee_{T \in \psi'_{\vec{z}}} \left(T \wedge y_1 \wedge y_2 \wedge t(\vec{z}) \right)$. Let $\vec{z} \in S$, and let $\psi_{\vec{z}}$ be the subformula of ψ consisting of those terms that are satisfied by $\vec{x}11\vec{z}$ for some $\vec{x} \in f^{-1}(1)$. Each term of $\psi_{\vec{z}}$ contains $y_1 \wedge y_2 \wedge t(\vec{z})$, since flipping either y_1 or y_2, or any bit of $\vec{z}$, produces an input that falsifies g'. It follows that $\psi_{\vec{z}} = \bigvee_{T \in \psi'_{\vec{z}}} \left(T \wedge y_1 \wedge y_2 \wedge t(\vec{z}) \right)$ where $\psi'_{\vec{z}}$ is a DNF. It remains to show that $\psi'_{\vec{z}}$ is consistent with f. For every $\vec{x} \in f^{-1}(1)$, there is a term of ψ that is satisfied by $\vec{x}11\vec{z}$, and thus there is a corresponding term of $\psi'_{\vec{z}}$ that is satisfied by $\vec{x}$. On the other hand, every $\vec{x} \in f^{-1}(0)$ must falsify $\psi'_{\vec{z}}$, as otherwise $\vec{x}11\vec{z}$ would satisfy ψ.

It follows from the above that ψ consists of the terms $t(\vec{x}) \wedge y_{2-p(\vec{x})}$ for each $\vec{x} \in f^{-1}(*)$; and of the subformulae $\psi_{\vec{z}}$, for each $\vec{z} \in S$. These components are pairwise disjoint. Since $\psi'_{\vec{z}}$ is consistent with f it follows that $\psi_{\vec{z}}$ contains at least $\textit{dnf-size}(f)$ terms, and thus the size of ψ is at least $s \cdot \textit{dnf-size}(f) + s$. □

The results of Section 4.2, together with Theorem 4.6, yield the following hardness result for *Min-DNF*.

THEOREM 4.8. *If* $NP \not\subseteq DTIME(2^{\mathrm{polylog}(n)})$ *then there exists a constant* $\gamma > 0$ *such that*

no polynomial time algorithm achieves an approximation ratio better than $(\log N)^\gamma$ *for Min-DNF, where N is the size of the truth table.*

4.3. An improved hardness result under additional assumptions. In this section, we prove an $\Omega(\sqrt{\log N})$ hardness of approximation result for *Min-DNF* under the additional assumption that a restriction of *Set Cover* is $\Omega(\log n)$-hard to approximate.

DEFINITION 4.9. *The f-Frequency Bounded Set Cover problem is the restriction of Set Cover to instances where each element occurs in at most $f(n)$ sets, where n is the total size of the instance.*

It is well-known ([21]) that a factor f approximation for f-*Frequency Bounded Set Cover* can be obtained in polynomial time. Thus for $f = o(\log n)$, f-*Frequency Bounded Set Cover* is not as hard to approximate as the general *Set Cover* problem. On the other hand, the reduction of Lund and Yannakakis showing an $\Omega(\log n)$ hardness of approximation for *Set Cover* produces an instance of $\Omega((\log n)^c)$-*Frequency-Bounded-Set-Cover*, for some constant c, which implies an $\Omega(\log n)$ hardness result for that problem. We conjecture that f-*Frequency-Bounded-Set-Cover* is NP-hard to approximate within a factor better than $c_2 \ln n$, for $f = c_1 \ln n$ and some constants c_1, c_2. Resolving this conjecture is an interesting question in its own right, since it postulates a frequency threshold (within a constant factor) for hardness. Assuming that the conjecture holds, we can prove an $\Omega(\sqrt{\log N})$ hardness of approximation result for *Min-DNF* using a simple, randomized reduction.

THEOREM 4.10. *If there exist constants c_1 and c_2 such that it is **NP**-hard to approximate $(c_1 \ln n)$-Frequency Bounded Set Cover to within $c_2 \ln n$, then there exists a constant c_3 such that no polynomial-time algorithm for Min-DNF achieves an approximation ratio better than $c_3 \sqrt{\log N}$ unless **NP** $\subseteq$ DTIME$(2^{\mathrm{polylog}(n)})$.*

Proof. Assume that there exist constants c_1 and c_2 as in the lemma. We prove an $\Omega(\sqrt{\log N})$ hardness of approximation for *Min-DNF(*)*; the reduction from section 4.2 extends the same result to *Min-DNF*. Let $(\mathscr{S}, \mathscr{U})$ be an instance of $(c_1 \ln n)$-*Frequency Bounded Set Cover* of size n. The idea of the reduction is as follows. First, we will map each set $S \in \mathscr{S}$ to a subset $f(S) \subseteq [b]$, for a suitably chosen parameter b. Second, we define vectors $w^S \in \{0,1\}^b$ for each $S \in \mathscr{S}$, by letting w^S have zeroes in those positions contained in $f(S)$ and ones elsewhere. Finally, we define vectors $u^x \in \{0,1\}^b$ for each $x \in \mathscr{U}$ having zeroes in the positions contained in F_x, where

$$F_x = \bigcup_{S \in \mathscr{S} : x \in S} f(S)$$

and ones elsewhere. If the vectors satisfy the condition $u^x \le w^S \iff x \in S$ for all $x \in \mathscr{U}$ and $S \in \mathscr{S}$, then by Lemma 3.1 we can construct an instance of *Min-DNF(*)* over b variables whose optimum is equal to the optimum for $(\mathscr{S}, \mathscr{U})$. Notice that the definition of u^x implies that the "if" part of the condition is always satisfied. For the "only if" part to hold, it is necessary and sufficient that for every S such that $x \notin S$, u_x has a one in a position where w^S has a zero; that is, $f(S) \not\subseteq F_x$. For $S \in \mathscr{S}$, let $f(S)$ be defined by choosing each $i \in [b]$ independently with probability p. Fix an element $x \in \mathscr{U}$, and a set $S \in \mathscr{S}$ such that $x \notin S$. We will show that the probability that $f(S) \subseteq F_x$ is small. For any choice of p, the probability that $f(S) \subseteq F_x$ is maximized when x occurs in exactly $c_1 \ln n$ sets (since it cannot occur in more than $c_1 \ln n$ sets). As we wish to find an upper bound for the probability that $f(S) \subseteq F_x$, we may therefore assume that x occurs in exactly $c_1 \ln n$ sets. For $1 \le i \le b$, let X_i be the indicator variable for the event $i \in F_x$. Then $\mathbf{E}[X_i] = 1 - (1-p)^{c_1 \ln n}$, and letting $X = \sum_{1 \le i \le b} X_i$ be the

size of F_x, linearity of expectation implies

$$\mathbf{E}[X] = b(1 - (1 - p)^{c_1 \ln n})$$
$$\approx b(1 - e^{-p c_1 \ln n})$$

which can be made smaller than $b/4$ by choosing $p \approx \ln(4/3)/(c_1 \ln n)$. The X_i's are independent, and we apply the simplified Chernoff bound $\Pr[X > (1 + \delta)\mathbf{E}[X]] < 2^{-\delta \mathbf{E}[X]}$ to obtain

$$\Pr[X > b/2] < 2^{-b/4}$$

Let us consider the case $|F_x| \leq b/2$. Then the probability that $f(S_j) \subseteq F_x$ is

$$\Pr\left[f(S_j) \subseteq F_x \,\middle|\, |F_x| \leq b/2\right] \leq (1 - p)^{b/2}$$
$$\approx \left(1 - \frac{\ln(4/3)}{c_1 \ln n}\right)^{b/2}$$
$$< e^{-b \ln(4/3)/(2 c_1 \ln n)}$$

Choosing $b = 8c_1 \ln^2 n$, we have

$$\Pr[f(S_j) \subseteq F_x] \leq 2^{-b/4} + e^{-4 \ln(4/3) \ln n}$$
$$< e^{-3 \ln n}$$
$$= 1/n^3$$

Applying the union bound, the probability that there exists an element $x \in \mathscr{U}$ and a set $S \in \mathscr{S}$ with $x \notin S$, such that $f(S) \subseteq F_x$, is at most $1/n$. Thus with probability at least $1 - 1/n$, we can apply the construction of Lemma 3.1 to the vectors u^x and w^S, to obtain an instance of *Min-DNF(*)* over $b = O(\log^2 n)$ variables whose minimum DNF has the same size as the minimum set cover for $(\mathscr{S}, \mathscr{U})$. The *Min-DNF(*)* instance has size $N = 2^b = O(n^{\log^2 n})$, and the probabilistic construction can be derandomized in quasi-polynomial time using the method of conditional probabilities (see, e.g., [6]). It follows that there is no polynomial-time algorithm for *Min-DNF(*)* which achieves an approximation ratio better than $c_2 \ln n = \Omega(\sqrt{\log N})$ unless $\mathbf{NP} \subseteq \mathrm{DTIME}(2^{\mathrm{polylog}(n)})$. □

5. A tight example for the greedy algorithm. We show that there exist instances of *Min-DNF* for which the greedy *Set Cover* algorithm achieves an $\Omega(\log N)$ approximation ratio. Our approach is to take a standard worst-case *Set Cover* instance and to apply a version of the reductions of Sections 3.1 and 4.2 to obtain first a *Min-DNF(*)* instance, and then a *Min-DNF* instance. We then show that the greedy algorithm operates on the resulting *Min-DNF(*)* instance, and on the resulting *Min-DNF* instance, much as it does on the original *Set Cover* instance.

5.1. Tight example for greedy on *Min-DNF(*)*. The starting point is the following *Set Cover* instance, on which the greedy *Set Cover* algorithm has worst-case behavior. The instance consists of $m - 1$ pairwise-disjoint sets $S_1, \ldots, S_{m-1}$, such that $|S_i| = 2^i$; and of two additional sets T_0 and T_1. For each set S_i, the set T_0 contains half of the elements in S_i, while T_1 contains the other half. On this set collection, the greedy algorithm chooses the cover consisting of all the sets S_i, while the optimal solution consists only of T_0 and T_1.

Let $\mathscr{U}$ be the underlying universe. We define three sets of vectors, $\{v^e \mid e \in \mathscr{U}\}$, $\{s^i \mid 1 \le i \le m-1\}$, and $\{t^0, t^1\}$, over $\{0,1\}^{2(m+1)}$, such that the following three conditions hold for all $e \in \mathscr{U}$ and all $1 \le i \le m-1$: (1) $v^e \le s^i$ iff $e \in S_i$; (2) $v^e \le t^0$ iff $e \in T_0$; and (3) $v^e \le t^1$ iff $e \in T_1$. The vectors $\{v^e \mid e \in \mathscr{U}\}$ are defined according to the set in which they occur, as follows: each element $e \in S_i$ is assigned a unique vector v^e from the set $\{x10^{m-i}\overline{x}01^{m-i} \mid x \in \{0,1\}^i\}$. The vectors $\{s^i \mid 1 \le i \le m-1\}$ and $\{t^0, t^1\}$ are defined as follows: $s^i = 1^i 10^{m-i} 1^i 01^{m-i}$; $t^0 = 01^m 1^{m+1}$, and $t^1 = 1^{m+1} 01^m$. The set T_0 is defined as $\{e \in \mathscr{U} \mid v^e \le t^0\}$, and the set T_1 is defined as $\{e \in \mathscr{U} \mid v^e \le t^1\}$. It is easily verified that the sets $S_1, \ldots, S_{m-1}, T_0, T_1$ have the required structure: namely, $S_1, \ldots, S_{m-1}$ are pairwise disjoint, S_i has size 2^i, and T_0 and T_1 are disjoint, each consisting of half of the elements from each of the sets $S_1, \ldots, S_{m-1}$. Conditions (2) and (3) hold by definition, as does the "if" direction of condition (1). For the "only if" direction of (1), note that if $e \in S_j$ for $j \ne i$, then either bit $j+1$ or bit $(m+1) + (i+1)$ witnesses the fact that $v^e \not\le s^i$.

The partial Boolean function f is defined as in the reduction from Section 3.1: the ones of f are the vectors v^e, for each $u \in \mathscr{U}$; the stars of f are those remaining inputs $\vec{x}$ such that $\vec{x} \le s^i$ for some $1 \le i \le m-1$, or $\vec{x} \le t^0$, or $\vec{x} \le t^1$; and the remaining inputs are zeroes. The following general lemma shows that the prime implicants of f, viewed as sets and considering only the ones of f that they cover, have exactly the same structure as the original set system.

LEMMA 5.1. *Let $\mathscr{S}$ be a set system over universe $\mathscr{U}$ such that no set in $\mathscr{S}$ contains another set in $\mathscr{S}$. Let $\{v^e \mid e \in \mathscr{U}\}$ and $\{w^S \mid S \in \mathscr{S}\}$ be sets of Boolean vectors such that the following condition holds for all $e \in \mathscr{U}$ and all $S \in \mathscr{S}$: $v^e \le w^S$ if and only if $e \in S$. Let f be the function obtained from $\mathscr{S}$ as in Lemma 3.1 using the given vectors. Then the set of prime implicants of f is exactly $\{\tau(w^S) \mid S \in \mathscr{S}\}$.*

Proof. We first show that each term $\tau(w^S)$ is, indeed, a prime implicant of f. Suppose, on the contrary, that there is an implicant τ of f that subsumes $\tau(w^S)$, for some $S \in \mathscr{S}$. Note that all variables in τ are negated. Let $\vec{u}$ be a maximal truth assignment satisfying τ. Since $f(\vec{u}) = 1$, there is a set $S' \in \mathscr{S}$ such that $\vec{u} \le w^{S'}$; that is, for each index i, if $w_i^{S'} = 0$ then $u_i = 0$. By our choice of $\vec{u}$ we have that $u_i = 0$ iff the literal $\neg x_i$ occurs in τ, and by definition $w_i^{S'} = 0$ iff the literal $\neg x_i$ occurs in $\tau(w_i^{S'})$. Thus for each index i, if the literal $\neg x_i$ occurs in $\tau(w^{S'})$ then it also occurs in τ. As both τ and $\tau(w^{S'})$ consist exclusively of negated literals, we have $\tau \implies \tau(w^{S'})$; and since τ subsumes $\tau(w^S)$ we have

$$\tau(w^S) \implies \tau \implies \tau(w^{S'})$$

For each $e \in U$, $e \in S \iff v^e \le w^S$, and $v^e \le w^S$ iff v^e satisfies $\tau(w^S)$. Thus,

$$e \in S \implies v^e \le w^S \implies v^e \le w^{S'} \implies e \in S'$$

That is, $S \subseteq S'$, contradicting the assumption about $\mathscr{S}$.

We now show that every prime implicant of f is equal to $\tau(w^S)$, for some $S \in \mathscr{S}$. Let τ be a prime implicant of f, and let $\vec{u}$ be a maximal truth assignment satisfying τ. Then there exists $S \in \mathscr{S}$ such that $\vec{u} \le w^S$, and thus $w_i^S = 0$ implies $u_i = 0$, and by the same argument as before we have that each literal $\neg x_i$ occurring in $\tau(w^S)$ also occurs in τ. It follows that $\tau = \tau(w^S)$. $\square$

Lemma 5.1 implies that the prime implicants of f, viewed as sets, have exactly the same structure with respect to the ones of f as the original set collection has with respect to $\mathscr{U}$. It follows that the greedy algorithm finds a solution of size $m-1$, consisting of the terms

$\tau(s^1) \wedge \ldots \wedge \tau(s^{m-1})$, while the optimal solution, $\tau(t^0) \wedge \tau(t^1)$, has size two. As the instance has $n = 2m + 2$ variables, the approximation ratio is $(m-1)/2 = (n-4)/4$.

5.2. Tight example for greedy on *Min-DNF*. We now extend the construction of the previous section to give an instance of *Min-DNF* for which the greedy algorithm achieves an approximation ratio of $\Omega(n) = \Omega(\log N)$. The instance is obtained by applying the reduction of Section 4.2 to the function f from Section 5.1. As in the proof of Lemma 4.7, we use $t(\vec{w})$ to denote the term $\left(\bigwedge_{i:w_i=1} w_i \right) \wedge \left(\bigwedge_{i:w_i=0} \neg w_i \right)$.

LEMMA 5.2. *Let $\mathcal{S}$ be a set of subsets of $[n]$, and let f be a partial Boolean function, such that $\mathcal{S}$ and f satisfy the conditions of Lemma 3.1. Let g be the total Boolean function obtained by applying the reduction from Section 4.2 to f. Then the prime implicants of g consist of exactly the following: $\tau \wedge y_1 \wedge y_2 \wedge t(\vec{z})$, where τ is a prime implicant of f and $\vec{z} \in S$, and $t(\vec{x}) \wedge y_{2-p(\vec{x})}$ where $f(\vec{x}) = *$.*

Proof. It is easy to verify that each term in the statement of the lemma is, indeed, a prime implicant of g, noting that every prime implicant τ of f must cover at least one vector in $f^{-1}(1)$, since each set in the original set system is non-empty.

We now argue that all prime implicants of g are of the above types. Let τ be an implicant of g: we will show that τ is subsumed by an implicant of one of the above two types. Let $\vec{x}y_1y_2\vec{z}$ be an assignment that satisfies τ. We first consider the case where $f(\vec{x}) = 1$. From the definition of g, it is clear that τ contains $t(\vec{z})$, y_1, and y_2, as flipping the corresponding bits of the assignment falsifies g. Moreover, the portion τ_x of τ containing x-variables is an implicant of f, and is therefore subsumed by a prime implicant τ' of f. Thus $\tau' \wedge y_1 \wedge y_2 \wedge t(\vec{z})$ is an implicant of g, and subsumes τ. Now consider the case where $f(\vec{x}) = *$, and assume without loss of generality that $\vec{x}$ has even parity. Then τ must contain y_2 and $t(\vec{x})$, as flipping any of these bits falsifies g, and thus τ is subsumed by $t(\vec{x}) \wedge y_2$. $\square$

The inputs to g are of the form $\vec{x}y_1y_2\vec{z}$ where the length of $\vec{x}$ is n, and the length of $\vec{z}$ is $t = n+1$. Each prime implicant of the second type in the statement of Lemma 5.2 covers 2^{t+1} ones of g and the ones covered by these prime implicants are pairwise disjoint. The prime implicants of the first type each cover at most $2^{n-1} < 2^{t+1}$ ones of g. Thus the greedy algorithm begins by choosing all prime implicants of the second type. At this point, the prime implicants of the first type corresponding to different values of $\vec{z}$ cover disjoint subsets of the ones of g, so let us only consider a particular value of $\vec{z}$: the choices made by the greedy algorithm for other vectors $\vec{z}$ are independent of its behaviour on this vector. Now the uncovered ones of g that are covered by a term $\tau \wedge y_1 \wedge y_2 \wedge t(\vec{z})$ are precisely those whose $\vec{x}$-component is a one of f, as the others are already covered by prime implicants of the second type. Thus the prime implicants of this type chosen by the greedy algorithm are exactly the set of prime implicants of the form $\tau \wedge y_1 \wedge y_2 \wedge t(\vec{z})$, where τ is a prime implicant that would be chosen by the greedy algorithm on input f. It follows that the greedy solution has size $s(m-1) + s = sm$, while the optimal solution has size $2s + s = 3s$. As the instance has $n = 2m + 4 + t$ variables, the approximation ratio is $m/3 = (n - t - 4)/6 = \Omega(n)$.

6. Fixed Parameter Complexity. It is known that the decision problem "Given a truth table of a Boolean function f, and a number k, does f have a DNF with at most k terms?" can be solved in time $p(N, 2^{k^2})$, for some polynomial p, where N is the size of the truth table [20]. (This follows easily from the fact that if f is a Boolean formula that can be represented by a k-term DNF formula, then there exist at most 2^k prime implicants of f [12].) Thus, *Min-DNF* is *fixed parameter tractable* [16]. Moreover, because the size of the input truth table

is $N = 2^n$, where n is the number of variables of f, it follows that *Min-DNF* is solvable in polynomial time for any $k = O(\sqrt{n})$.

It is an open question whether *Min-DNF* can be solved in polynomial time for $k = n$. But by applying a simple padding argument, we obtain the following corollary to the NP-completeness result for *Min-DNF*:

COROLLARY 6.1. *If there exists some constant $\varepsilon > 0$ such that NP is not contained in $DTIME(2^{O(n^\varepsilon)})$, then for some constant $c > 1$, Min-DNF for $k = n^c$ is not solvable in polynomial time (where n is the number of input variables of the Boolean function defined by the Min-DNF instance).*

Proof. Because *Min-DNF* is NP-complete, there exists a polynomial-time reduction from problems Π in NP to *Min-DNF*. If the input to Π is of size n, then the input to the resulting *Min-DNF* problem will be a truth table of size $s = O(n^b)$ for some constant $b > 1$, defining a Boolean function on $\log s$ variables. The parameter k in the derived *Min-DNF* instance is no more than s, since for any truth table, there is always a consistent DNF of size at most the size of the truth table. Let $c > 1$. Let $m = s^{\frac{1}{c}}$. Take the *Min-DNF* instance and form a new *Min-DNF* instance by padding the function in the truth table with $m - \log s$ new dummy variables. Suppose *Min-DNF* is solvable in polynomial time when $k = n^c$, where n is the number of input variables of the Boolean function defined by the *Min-DNF* instance. Then the padded instance of *Min-DNF* can be solved in time polynomial in 2^m, and Π can be solved in time $2^{O(n^{\frac{b}{c}})}$, where n is the size of input to Π. For $c > \frac{b}{\varepsilon}$, this is less than $2^{O(n^\varepsilon)}$. Contradiction. $\square$

7. *Min-DNF* and learning. One of the major problems in learning theory is to determine whether DNF formulas can be learned in polynomial time. There are connections between the complexity of *Min-DNF* and its fixed parameter versions, and the complexity of learning DNF formulas. This connection is strongest for "proper" learning models. In such models, any hypotheses used in the learning algorithm must be of the same type as the formulas being learned by the algorithm. Thus if the task is to learn DNF formulas, hypotheses must be DNF formulas. If the task is to learn k-term DNF formulas, then hypotheses must be k-term DNF formulas.

There has been a significant amount of research on learning k-term DNF formulas for small values of k in both proper and improper models (see e.g. [8, 10, 28, 20, 35]). Pitt and Valiant showed that in the PAC model, unless RP=NP you cannot learn k-term DNF formulas in polynomial time using hypotheses that are k-term DNF formulas (for constant k) [35]. Their proof actually shows that the consistency problem for k-term DNF is hard. This problem takes as input a partial Boolean function, specified by its 1's and 0's, and asks whether there is a k-term DNF formula consistent with those entries. The work of Pitt and Valiant was subsequently extended to obtain significantly stronger results on learning arbitrary length DNF formulas in the PAC learning model [2, 18]. We note that our reduction to *Min-DNF(*)* in fact implies that the consistency problem for k-term DNF is NP-hard for $k = n$, even when the underlying function depends on only $\log n$ of the n input variables (a $\log n$ "junta"); this in turn implies that proper PAC learning of n-term DNF formulas depending on $\log n$ of the n input variables is hard unless RP=NP.

Pillaipakkamnatt and Raghavan [34] showed that for some $\varepsilon < 1$ and some $c > 1$, $\log^c n$-term DNF cannot be learned in the membership and proper equivalence query model unless $NP \subseteq DTIME(2^{O(n^\varepsilon)})$. Subsequently, Hellerstein and Raghavan proved that $\Omega(\log^{3+\varepsilon} n)$-term DNF formulas cannot be learned in the same model; their proof involves a structural property of DNF formulas and the result is without any assumptions [20]. (It can be improved to $\Omega(\log^{2+\varepsilon})$.) It is open, however, whether $\log n$-term DNF formulas can be learned in

polynomial time in this model; $\sqrt{\log n}$-term DNF can be so learned [10].

A polynomial-time algorithm for learning $\log n$-term DNF formulas in the membership and proper equivalence query model (i.e. with hypotheses that are $\log n$-term DNF formulas) would imply a polynomial-time algorithm for *Min-DNF* for $k = n$ [20]. The same proof shows that for constant c, a polynomial-time algorithm for learning $\log^c n$-term DNF formulas would imply a polynomial-time algorithm for *Min-DNF* for $k = n^c$. It follows that the result of [34] mentioned above can also be derived from Corollary 6.1.

The relation between truth table minimization and learning with membership and equivalence queries relies on the following observation: Given a truth table representing a function f, one can simulate a membership and equivalence query algorithm for learning (a hypothesis representing) f by using the truth table to answer the queries. Feldman observed that one can also use the truth table of f to generate uniformly distributed examples of f. Combining this observation with the hardness of approximating *Min-DNF* he showed hardness of proper PAC learning of min-DNF under the uniform distribution, with membership queries. More specifically, he showed that for some $\gamma > 0$, unless P=NP, there is no polynomial-time algorithm that PAC learns DNF formulas under the uniform distribution using hypotheses that are DNF formulas of size at most n^γ larger than the function being learned, even if membership queries are allowed [18].

8. Hardness of *Min-AC_d^0*. In [4] it was shown that neither *Min-Circuit* nor *Min-NC^1* can be approximated to within a factor of $n^{1-\varepsilon}$ in polynomial time unless Blum Integers can be factored efficiently. Here we strengthen that result to hold for *Min-AC^0* also. More precisely, let *Min-AC_d^0* be the problem of estimating the size of the smallest depth-d AC^0 circuit for a function, given its truth table. In this section, we show that for large enough d, *Min-AC_d^0* is hard to estimate, unless Blum integers can be factored in time 2^{n^δ}, where the value of d depends on the value of δ. We have not computed the relationship between d and δ, but we anticipate that this yields a meaningful inapproximability result for d as small as 10.

LEMMA 8.1. *For every language $\mathscr{L}$ in NL, and for every ε, there exists a d such that there are AC_d^0 circuits of size 2^{n^ε} that recognize $\mathscr{L}$.*

Proof. Consider NL machines running in time $m = n^c$. To find an accepting path, guess $\sqrt{m}$ "checkpoints" and verify that, for every 2 adjacent "checkpoints" there is a path of length $\sqrt{m}$ connecting them. This can be implemented by a depth-three AC^0 circuit of size $2^{O(\sqrt{m}\log m)}$. If the number of checkpoints chosen is $m^{1/3}$, then a similar strategy leads to a depth-five circuit of size $2^{O(m^{1/3}\log m)}$. That is, the top level of the depth-five circuit is an OR over all the $2^{O(m^{1/3}\log m)}$ sequences of checkpoints, of the AND that each of the $m^{1/3} - 1$ pairs of adjacent checkpoints is connected by a path of length $m^{2/3}$. This latter condition can be checked by an OR over another $2^{O(m^{1/3}\log m)}$ sequences of $m^{1/3}$ checkpoints, of an AND that each of the $m^{1/3} - 1$ pairs of adjacent checkpoints is connected by a path of length $m^{1/3}$. Since the input head can only move a distance of $m^{1/3}$ in $m^{1/3}$ steps, and each checkpoint specifies the position of the input head, the condition that a given pair of checkpoints is connected by a path of length $m^{1/3}$ depends only on $m^{1/3}$ input variables, namely those centered around the two input head positions specified by the checkpoints. Thus this condition can be expressed by a CNF formula of size exponential in $m^{1/3}$. (The depth can be optimized somewhat, using closure under complement and merging adjacent layers – but we ignore such issues for now.)

Iterating the above idea gives depth-d AC^0 circuits of size 2^{n^ε}. This is basically a strengthening of Nepomnjaščiĭ's Theorem [5, 33]. (The same claim, with an identical proof, holds for any language accepted by a nondeterministic machine running in polynomial time and using space $n^{o(1)}$. In particular, it holds for the complexity class LogCFL.) □

DEFINITION 8.2. *A 2n-bit integer is called a Blum Integer if $N = PQ$, where P and Q are two primes such that $P \equiv Q \equiv 3 \mod 4$. The Blum Integer Factorization problem is as follows. Given a Blum Integer N find the primes P and Q such that $1 < P \leq Q$ and $N = PQ$.*

THEOREM 8.3. *For every $\delta > 0$ and $\varepsilon > 0$ there is a depth d such that Blum Integer Factorization is in $BPTIME(2^{n^{\varepsilon}})^{\mathscr{B}}$, where $\mathscr{B}$ is any function that approximates Min-AC_d^0 to within a factor of $n^{1-\delta}$ (here n is the size of the truth-table input to Min-AC_d^0).*

Proof. We follow the proof given in [4]. In [32] a pseudo-random function ensemble $\{f_{N,r}(x) : \{0,1\}^n \to \{0,1\}\}_{N,r}$ is constructed with the following two properties:

- There is a TC^0 circuit computing $f_{N,r}(x)$, given a $2n$ bit integer N, a $4n^2 + 2n$-bit string r, and an n-bit string x.
- For every probabilistic Turing machine $\mathscr{M}$ running in time $t(n)$ with oracle access to $f_{N,r}$ of query length n, there exists a probabilistic Turing machine $\mathscr{A}$ running in time $t(n)n^{O(1)}$ such that for every $2n$-bit Blum integer $N = PQ$, if $|Pr[\mathscr{M}^{f_{N,r}}(N) = 1] - Pr[\mathscr{M}^{R_n}(N) = 1]| > 1/2$, where R_n is a uniformly distributed random function ensemble, and the probability is taken over random r, and random bits of $\mathscr{M}$, then $Pr[\mathscr{A}(N) = (P,Q)] > 1/2$. In other words, if $\mathscr{M}$ can distinguish the pseudorandom function ensemble from a truly random function "efficiently", then Blum Integers can be factored "efficiently" on a probabilistic machine.

Our pseudrandom function ensemble $f_{N,r}$ is computable by a TC^0 circuit of size $n^{c'}$ for some constant c'. Let $x_1, x_2, \ldots, x_{2^n}$ denote the strings in $\{0,1\}^n$ in lexicographic order. Let m be a power of two of size approximately $2^{n^{\varepsilon}}$. For all large enough n, all $2n$-bit integers N and all $4n^2 + 2n$-bit strings r, consider the function $h_{N,r}$ whose truth table is given by $f_{N,r}(x_1), f_{N,r}(x_2), \ldots, f_{N,r}(x_m)$. This function has TC^0 circuits of size polynomial in n and hence by Lemma 8.1, there is some d (depending only on ε) such that this function has depth-d AC^0 circuits of size $2^{n^{\gamma}}$ for some $\gamma < \varepsilon$.

Now consider the following oracle Turing machine $\mathscr{M}$ with access to oracle $\mathscr{B}$ and to an oracle g: On input N, $\mathscr{M}$ queries g on the lexicographically first m inputs, $x_1, \ldots, x_m$, to get answers $y_1, \ldots, y_m$; let h denote the function whose truth table is given by $y_1, \ldots, y_m$. $\mathscr{M}$ then submits h to the approximation algorithm $\mathscr{B}$. If $\mathscr{B}$ says that h has AC_d^0 circuits of size less than $m^{1-\delta/2}$ then M will reject; otherwise M will accept. We are assuming that $\mathscr{B}$ approximates Min-AC_d^0 within a factor of $m^{1-\delta}$, i.e., that $1 \leq \mathscr{B}(h)/Min\text{-}AC_d^0(h) \leq m^{1-\delta}$. Now for sufficiently large n, if the oracle g is chosen from the pseudorandom distribution $\{f_{N,r}\}_{N,r}$ then $\mathscr{M}$ will always reject, since as argued above, in this case $h = h_{N,r}$ has AC_d^0 circuits of size at most $2^{n^{\gamma}}$, and $2^{n^{\gamma}} m^{1-\delta} < m^{1-\delta/2}$. On the other hand, if g is taken uniformly at random from $\mathscr{R}_n$, then $y_1 \ldots y_m$ is a random string, and thus with extremely high probability the function represented by $y_1 \ldots y_m$ requires AC_d^0 circuits of size $m/\log m > m^{1-\delta/2}$ (since most functions require circuits of this size), and this condition causes $\mathscr{M}$ to accept. Hence $|Pr[\mathscr{M}^{f_{N,r}(x)}(N) = 1] - Pr[\mathscr{M}^{R_n}(N) = 1]| > 1/2$, for sufficiently large n. Thus, $\mathscr{M}$ can distinguish the pseudorandom function ensemble from a truly random one with probability greater than $1/2$, and thus Blum Integers can be efficiently factored probabilistically. $\square$

COROLLARY 8.4. *For all $\delta > 0$ and all $\varepsilon > 0$ there exists a d such that Min-AC_d^0 cannot be approximated to within a factor $n^{1-\delta}$ in BPP unless Blum Integer Factorization is in $BPTIME(2^{n^{\varepsilon}})$.*

9. Discussion. There are close connections between the hardness of function minimization problems and related learnability results. In addition to the connections discussed above in Section 6, we mention two others: the complexity of *Min-DNF*(DNF) and approximating *Min-DNF*(DNF) has been shown to be related to the problem of learning DNF with proper membership and equivalence queries [11, 20, 1], and results on learning circuits [9] yield positive results for approximating circuit minimization (cf. [42]). At a basic level, learning a formula or circuit involves gathering information about it, and then synthesizing or compressing that information to produce a compact hypothesis. The need for compactness provides the connection to minimization. In many learning problems one can distinguish between informational complexity (the number of queries or sample size needed), and computational complexity (the amount of computation needed to process the information). Information about a formula or circuit typically consists just of input/output pairs. Truth table minimization problems are relevant to the computational hardness of learning; even if you have all input/output pairs, the question is whether you can compact that information in polynomial time.

The NP-hardness of proper PAC learning DNF and of *Min-DNF* are known. On the other hand, very strong inapproximability results are known for both proper PAC learning and the function minimization problem for complexity classes starting at AC^0. However, these latter results rely on cryptographic assumptions, and are not known to hold under NP-hardness assumptions. Thus an important open question is to resolve the NP-hardness of both learnability results as well as function minimization results above for classes that are stronger than DNF.

Another open problem is to close the approximability gap for *Min-DNF*.

REFERENCES

[1] A. Aizenstein, T. Hegedus, L. Hellerstein, and L. Pitt, "Complexity theoretic hardness results for query learning", *Computational Complexity* 7(1), 1998, pp. 19-53.

[2] M. Alekhnovich, M. Braverman, V. Feldman, A. Klivans, and T. Pitassi, "Automatizability and Learnability", in Proceedings of the *45th Annual IEEE Symposium on Foundations of Computer Science*, Rome, Italy, 2004.

[3] E. Allender, L. Hellerstein, P. McCabe, T. Pitassi, and M. Saks, "Minimizing DNF Formulas and AC^0 Circuits Given a Truth Table", *ECCC TR05-126*, 2005. Also appeared in Proceedings of the *21st Annual IEEE Conference on Computational Complexity*, 2006.

[4] E. Allender, M. Koucky, D. Ronneberger, and S. Roy, "Derandomization and distinguishing complexity", in Proceedings of the *18th Annual IEEE Conference on Computational Complexity*, Aarhus, Denmark, 2003, pp. 209-220.

[5] E. Allender, M. Koucky, D. Ronneburger, and S. Roy, "Time-space tradeoffs in the counting hierarchy", in Proceedings of the *16th Annual IEEE Conference on Computational Complexity*, Chicago, Illinois, USA, 2001, pp.295-302.

[6] N. Alon, J. Spencer, and P. Erdös, *The Probabilistic Method*, John Wiley & Sons, 1992.

[7] N. Alon, D. Moshkovitz, and S. Safra, "Algorithmic Construction of Sets for *k*-Restrictions", to appear in ACM Transactions on Algorithms.

[8] U. Berggren, "Linear time deterministic learning of k-term DNF", in Proceedings of the *6th Annual ACM Conference on Computational Learning Theory*, Santa Cruz, California, USA, 1993, pp. 37-40.

[9] N. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, and C. Tamon, "Oracles and Queries That Are Sufficient for Exact Learning", *Journal of Computer and System Sciences* 52(3), 1996, pp.

421-433.

[10] N. H. Bshouty, "Simple learning algorithms using divide and conquer", *Computational Complexity* 6, 1997, pp. 174-194.

[11] N. H. Bshouty and L. Burroughs, "On the proper learning of axis parallel concepts", in Proceedings of the *15th Annual Conference on Computational Learning Theory*, Sydney, Australia, 2002, pp. 287-302.

[12] A. Chandra and G. Markowsky, "On the number of prime implicants", *Discrete Mathematics* 24, 1978, pp. 7-11.

[13] V. Chvátal, "A greedy heuristic for the set covering problem", *Mathematics of Operations Research* 4, 1979, pp. 233-235.

[14] O. Coudert, "Two-level logic minimization: an overview", *Integration, the VLSI Journal* 17(2), 1994, pp. 97-140.

[15] Sabastian Czort, *The complexity of minimizing disjunctive normal form formulas*, Master's Thesis, University of Aarhus, 1999.

[16] R. Downey and M. Fellows, *Parameterized Complexity*, Springer Verlag, 1999.

[17] U. Feige, "A threshold of ln n for approximating set cover", *Journal of the ACM* 45(4), 1998, pp. 634-652.

[18] V. Feldman, "Hardness of Approximate Two-level Logic Minimization and PAC Learning with Membership Queries", ECCC TR05-127, 2005. Extended version to appear in Proceedings of the *38th Annual ACM Symposium on the Theory of Computing*, 2006.

[19] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[20] L. Hellerstein and V. Raghavan, "Exact learning of DNF formulas using DNF hypothesis", *Journal of Computer and System Sciences* 70(4), 2005, pp. 435-470.

[21] D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11:555-556, 1982.

[22] D. S. Johnson, "Approximation algorithms for combinatorial problems", *Journal of Computer and System Sciences* 9, 1974, pp. 256-278.

[23] V. Kabanets, "Easiness assumptions and hardness tests: Trading time for zero error", *Journal of Computer and System Sciences* 63(2), 2001, pp. 236-252.

[24] V. Kabanets and J. L. Cai, "Circuit minimization problem", in Proceedings of the *32nd Annual ACM Symposium on the Theory of Computing*, Portland, Oregon, USA, 2000, pp. 73-79.

[25] M. Kearns and L. Valiant, "Cryptographic limitations on learning boolean formula and finite automata", *Journal of the ACM* 41(1), 1989, pp. 433-444.

[26] S. Khot, Online lecture notes for *Probabilistically Checkable Proofs and Hardness of Approximation*, Lecture 3 (scribed by Deeparnab Chakrabarti), available at `www.cc.gatech.edu/~khot/pcp-course.html`.

[27] K. Krajíček and P. Pudlák, "Some consequences of cryptographic conjectures for S_2^1 and EF", In *Logic and Computational Complexity: International Workshop, LCC '94*, volume 960 of *Lecture Notes in Computer Science*, Springer-Verlag, 1995, pp. 210-220.

[28] E. Kushilevitz, "A simple algorithm for learning $o(\log n)$-term DNF", *Information Processing Letters* 61(6), 1997, pp.289-292.

[29] L. Lovász, "On the ratio of optimal integral and fractional covers", *Discrete Mathematics* 13, 1975, pp. 383-390.

[30] C. Lund and M. Yannakakis, "On the hardness of approximating minimization problems", *Journal of the ACM* 41(5), 1994, pp. 960-981.

[31] W. J. Masek, "Some NP-complete set covering problems", Unpublished Manuscript, 1979.

[32] M. Naor and O. Reingold, "Number-theoretic constructions of efficient pseudorandom functions", In Proceedings of the *38th Annual IEEE Symposium on Foundations of Computer Science*, Miami Beach, Florida, USA, 1997, pp. 458-467.

[33] V. A. Nepomnjaščiĭ, "Rudimentary predicates and Turing calculations" *Math. Dokl* 11, 1970, pp. 1462-1465.

[34] K. Pillaipakkamnatt, and V. Raghavan, "On the Limits of Proper Learnability of Subclasses of DNF formulas", *Machine Learning* 25(2-3), 1996, pp. 237-263.

[35] L. Pitt and L. G.Valiant, "Computational limitations on learning from examples", *Journal of the ACM* 35, 1988, pp. 965-984.

[36] R. Raz, "A Parallel Repetition Theorem", *SIAM Journal on Computing* 27, 1998, pp. 763-803.

[37] A. A. Razborov and S. Rudich, "Natural proofs", *Journal of Computer and System Sciences* 55(1), August 1997, pp. 24-35.

[38] I. Dinur, E. Fischer, G. Kindler, R. Raz, S. Safra, PCP characterization of NP: Towards a Polynomially Small Error Probability, In Proceedings of the *31st Annual ACM Symposium on Theory of Computing*, Atlanta, Georgia, USA, 1999, pp. 29-40.

[39] L. Trevisan, "Non-approximability results for optimization problems on bounded degree instances", In Proceedings of the *33rd Annual ACM Symposium on Theory of Computing*, Heraklion, Crete, Greece, 2001, pp. 453-461.

[40] Larry J. Stockmeyer, "The polynomial-time hierarchy", *Theoretical Computer Science* 3, 1977, pp. 1-22.

[41] C. Umans, "The minimum equivalent DNF problem and shortest implicants", In Proceedings of the *39th Annual IEEE Symposium on Foundations of Computer Science*, Palo Alto, California, USA, 1998, pp. 556-563.

[42] C. Umans, "Hardness of approximating Σ_2^p minimization problems", In Proceedings of the *40th Annual IEEE Symposium on Foundations of Computer Science*, New York City, New York, USA, 1999, pp. 465-474.

[43] C. Umans, "Complexity of Two-Level Logic Minimization", *Technical Report No. UCB/ERL M04/45*, Berkeley, CA, October 2004.

Appendix

Appendix A. Reduction from *r-Uniform Set Cover*.

The following lemma describes a modified version of the reduction given in Section 3.1. Whereas the reduction in that section is from *3-Partite Set Cover*, the reduction here is from *r-Uniform Set Cover* (all sets in the input set cover instance are of size *r*). Because the reduction here is not from a partite version of *Set Cover*, it requires different techniques than the reduction in Section 3.1.

LEMMA A.1. *There is an algorithm takes as input an r-uniform collection of subsets $\mathscr{S}$ over $[n]$, and produces the truth table of a partial Boolean function f such that the minimum size of a cover of $[n]$ with $\mathscr{S}$ is equal to the minimum number of terms in a DNF consistent with f. The algorithm runs in time $(n|\mathscr{S}|)^{O(r)}$ and the number of variables of f is $O(r\log(n|\mathscr{S}|))$.*

Proof. Let the *r*-uniform collection $\mathscr{S}$ over $[n]$ be given.

As in the proof of Lemma 3.2, we produce two indexed sets of vectors $V = \{v^i : i \in [n]\}$ and $W = \{w^A : A \in \mathscr{S}\}$ of length *t* satisfying the property $(*)$ that for all $A \in \mathscr{S}$ and $i \in [n]$, $i \in A$ if and only if $v^i \leq w^A$. Again, we specify *V* and then define *W* according to the rule that for $A \in \mathscr{S}$, w^A is the bitwise OR of $\{v^i : i \in A\}$. The construction of partial function *f*, given *V* and *W*, is then the same as in the proof of Lemma 3.2, and again it follows that the size of the minimum DNF consistent with *f* is equal to the size of the minimum cover of $[n]$ by $\mathscr{S}$.

We now describe the construction of *V*. Let *P* be the set of pairs (j, A) with $A \in \mathscr{S}$ and $j \in [n] - A$. The desired conditions on *V* can be restated as specifying that for all $(j, A) \in P$:

$C(j, A)$: There is a bit position $\alpha \in [t]$ such that $v_\alpha^j = 1$ and $v_\alpha^i = 0$ for all $i \in A$.

If we choose $v^1, \ldots, v^n$ of length *t* at random where each bit is 1 independently with probability $1/r$, then for each $(j, A) \in P$ the probability that $C(j, A)$ does not hold is $(1 - \frac{1}{r}(1 - \frac{1}{r})^r)^t \leq e^{-t/3r}$, so the probability that $v^1, \ldots, v^r$ fails to meet the requirements is at most $|P|e^{-t/3r} \leq |\mathscr{S}|ne^{-t/3r}$. Thus if $t \geq 3r(1 + \ln(|\mathscr{S}|n))$ this random choice succeeds with

probability more than 1/2. This is enough for a randomized reduction. To make it deterministic, we derandomize this construction using the method of conditional probabilities (see, e.g., [6]). This is routine but technical so we provide only a sketch. Let $X(j,A)$ be the random variable that is 1 if $C(j,A)$ fails. We want to choose $v^1,\ldots,v^r$ so that $X = \sum_{(j,A)\in P} X(j,A) = 0$. The above argument says that under random choice $\mathbf{Exp}[X] \le 1/2$. The key point for derandomizing is that if we fix any subset of the bits in $v^1,\ldots,v^r$ then it is straightforward to compute the conditional expectation of X given this fixing in time $(|\mathscr{S}|n)^{O(1)}$. We can then use the method of conditional probabilities to fix these bits one at a time always choosing the value of the bit that does not increase the expectation. Once all bits are fixed we must have a good choice for V.

Clearly V and W can be constructed in time in time $(n|\mathscr{S}|)^{O(1)}$ with $t = O(r\log(n|\mathscr{S}|))$. Since the truth table has size 2^t, outputting it takes time $(n|\mathscr{S}|)^{O(r)}$. $\square$

Combining the above reduction with the modified reduction from *Min-DNF(*)* to *Min-DNF* in Section 4.2 yields a reduction from *r-Uniform Set Cover* to *Min-DNF*. Setting $r = \log N$, where N is the truth table size, one can then apply inapproximability results for *r-Uniform Set Cover* [17, 39] to show that *Min-DNF* cannot be approximated to within a factor of $\Omega(\log\log N)$ in polynomial time unless NP is in randomized quasipolynomial time.