Electronic Colloquium on Computational Complexity, Report No. 126 (2005)

Minimizing DNF formulas and AC^0_d circuits given a truth table

Eric Allender
Rutgers University

Lisa Hellerstein
Polytechnic University

Paul McCabe
University of Toronto

Toniann Pitassi
University of Toronto

Michael Saks
Rutgers University

November 4, 2005

Abstract

For circuit classes R, the fundamental computational problem *Min-R* asks for the 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 begin by presenting a new reduction proving that *Min-DNF* is NP-complete. It is significantly simpler than the known reduction of Masek [21], which is from Circuit-SAT. We then present a more complex, approximation-preserving reduction, that yields new inapproximability results for *Min-DNF* . Finally, we extend known hardness results for *Min-*TC^0_d to obtain new hardness results for *Min-*AC^0_d , under cryptographic assumptions.

1 Introduction

A fundamental computational problem is to determine the minimum size of a boolean function in some representation class, given a truth table for that class. The following are two prominent examples: (1) *Min-DNF* which asks whether a given boolean function presented as a truth table has a k-term DNF and (2) *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 also obtain a hierarchy of natural problems between *Min-DNF* and *Min-Circuit* , including problems such 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 sophisticated state-of-the-art software packages.

ISSN 1433-8092

In the 1970's, Masek showed that *Min-DNF* is NP-complete [21]. Masek's result was cited by Garey and Johnson [12] and is widely known, but the proof was never published. Later, Czort presented a modernized, more readable version of Masek's proof [10]. 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 3-Dimensional 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 truth table. This suggests 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 [10]. By building on our simple NP-completeness proof for *Min-DNF* , we obtain a more complicated reduction (also from a restricted version of set cover) which allows us to prove a new inapproximability results for *Min-DNF* . In particular, we prove that if NP is not contained in quasipolynomial time, then *Min-DNF* cannot be approximated to within a factor of $c(\log N)^\gamma$ for some constants $c > 1$, $0 < \gamma \leq 1$ where N is the size of the input truth table.

Although the general *Min-DNF* problem is NP-hard, for $k = O(\sqrt{logn})$ *Min-DNF* is tractable [13]. 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 [13]; 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^0_d for all sufficiently large d. Under cryptographic assumptions, it is known that *Min-Circuit* , *Min-*NC^1 and *Min-*TC^0_d are not polynomial-time solvable [2]. (Nothing is stated explicitly in [2] regarding *Min-*TC^0_d , but it is implicit.) We extend the hardness results for *Min-*TC^0_d to obtain new hardness results for *Min-*AC^0_d as well, under cryptographic assumptions. This still leaves open the interesting question of whether or not *Min-Circuit* (or the other problems) are NP-complete. Kabanets and Cai [16] 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 Sections 4 and 5 we present the more complicated reduction and the associated non-approximability results. Section 6 concerns the fixed parameter versions of *Min-DNF* . Our hardness results for *Min-*AC^0_d appear in Section 7. Finally we conclude in Section 8 with open problems and future directions.

2 Preliminaries

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 $N = 2^n$. We define the size of a DNF formula to be the number of terms in it, and

thus a smallest DNF is one with the minimum possible number of terms.

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 four variations, depending on the input representation: (i) the A version, *Min-DNF(A)* ; (ii) the *Min-DNF* (tt) version; (iii) the (A, B) version, *Min-DNF(A,B)* ; and (iv) the *Min-DNF(*)* version.

- A version: 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* version: 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.

- (A, B) version: $A, B \subseteq \{0,1\}^n$ is 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 $b \in B$. Note that the input in this case specifies a partial function and we desire a minimum DNF that is consistent with the partial function.

- *Min-DNF(*)* version: 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 definitions for the first three versions are from [10], who gives an excellent survey of previous work on these problems.

The decision versions of the above problems ask, given the function, and a natural number k, whether or not there is a DNF formula realizing the function 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.

However, it appears to be more difficult to prove hardness results for *Min-DNF* than for the other versions. Because the input to *Min-DNF* is a truth table, "polynomial-time" for this version means polynomial in 2^n, where n is the number of variables of the function defined by the truth table. Also, because the input to *Min-DNF* is a total, rather than a partial, function, there is relatively little flexibility available when mapping an instance of another problem into an instance of *Min-DNF* .

As explained by Czort [10], 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 [24]. There is also a clean NP-hardness proof of the A version due to Gimpel (cf. [10]). Subsequently, Masek [21] 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 ratio n^{ϵ}. Unfortunately, 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 are covered by the prime implicant). Thus given all of the prime implicants, finding a smallest DNF is equivalent to finding a smallest cover for these prime implicant sets. Thus, as mentioned above, *Min-DNF* can be viewed as a special "hypercube" case of set cover, where the sets to be covered have a special form; namely they correspond to mini-cubes in the n-dimensional boolean cube. Applying the greedy algorithm for set cover it follows that *Min-DNF* can be approximated to within a factor of n, where n is the number of variables of the underlying function. Because of this close relationship between set cover and *Min-DNF* , we were motivated to try and uncover the underlying complexity and approximability of *Min-DNF* , and to unify previous known results on DNF minimization by viewing *Min-DNF* as a set cover problem.

In what follows, we deal not only with the standard set cover problem, but also with two restricted versions of it, r-Uniform Set Cover and r-Partite Set Cover.

Uniform Set Cover problem: On input $(n, m, r, \mathcal{S})$ where n, m and r are positive integers and $\mathcal{S}$ is an r-uniform collection over $[n]$, is there a subcollection $\mathcal{C}$ of size at most m whose union is $[n]$?

Partite Set Cover problem: On input $(n, m, r, \Pi, \mathcal{S})$ where n, m and r are positive integers, Π is an r-partition of $[n]$, and $\mathcal{S}$ is an r-partite set collection over Π, is there a subcollection $\mathcal{C}$ of size at most m whose union is $[n]$?

The version of 3-Uniform Set Cover where m is fixed to be $n/3$ is called Exact Cover by 3-Sets (XC3). The version of 3-Partite Set Cover where m is fixed to be $n/3$ is called 3D-Matching. XC3 and 3D-Matching are NP-complete, as are r-uniform and r-partite set cover for any $r \geq 3$ [12].

3 Simple proof that *Min-DNF* is NP-complete

Our new proof that *Min-DNF* is NP-complete is a modification of an early proof of Gimpel that was used to show the NP-completeness of the A version of DNF minimization.

We start by briefly describing Gimpel's reduction. It can be viewed as consisting of two phases. In the first phase we map an instance of X3C, or more generally, an instance of 3-Uniform Set Cover, to a partial function f as follows. Given input instance S of 3-Uniform Set Cover over n ground elements, we map both the sets as well as the ground elements to truth assignments over a set V, $|V| = n$, such that a set covers a ground element in S if and only if the assignment corresponding to the ground element is below (in the boolean lattice) the assignment corresponding to the set. Each ground element x_i, $i \leq n$ is mapped to the truth assignment that is all zero except for bit i which is 1. Each set maps to the truth assignment corresponding to the characteristic function of the set. The 1's of f are those truth assignments corresponding to ground elements; the stars of f are those truth 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 3-Uniform Set Cover.

In the second phase of Gimpel's reduction, we map the 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. We give the details of g below in Section 3.2. Since g is a function over $n+2$ variables, its truth table size is exponential in the size of the 3-Uniform Set Cover instance from the first phase. Thus Gimpel's reduction does not give a hardness result for *Min-DNF* . It does, however, give a hardness result for the A version of DNF minimization, because the number of 1's of g is polynomial in the size of the input 3-Uniform Set Cover instance (this would not necessarily be the case if the input were an arbitrary set cover instance, rather than an r-uniform set cover instance).

Our proof 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 start with an instance of 3D-Matching, or more generally, with any instance of 3-Partite Set Cover, rather than from an instance of 3-Uniform Set Cover. The second phase of our reduction is identical to Gimpel's (except that we use truth table representations of f and g, and Gimpel does not).

3.1 First phase of the *Min-DNF* NP-completeness reduction

In the first phase of our reduction, we reduce 3-Partite Set Cover to *Min-DNF(*)* . The reduction is described by the following lemma.

Lemma 1. *There is an algorithm that takes as input an instance $(n, m, 3, \Pi, \mathcal{S})$ 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 $(n, m, 3, \Pi, \mathcal{S})$ of 3-Partite Set Cover, the algorithm produces an indexed set of vectors $V = \{v^i : i \in [n]\}$ and $W = \{w^A : A \in \mathcal{S}\}$ all of the same (small) length t satisfying:

(*) For all $A \in \mathcal{S}$ and $i \in [n]$, $i \in A$ if and only if $v^i \leq w^A$,

where the ordering on vectors is bitwise.

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

Let $q = \lceil \log_2 n \rceil$. For $i \in [n]$ write $\Pi(i)$ for the index of the block of Π that contains i and let $b(i)$ be the $2q$-bit string consisting of the q-bit binary expansion of i followed by its bitwise complement. Let $t = 6\lceil \log_2 n \rceil$. We will consider the the t-bit strings in V and W as being divided into 3 blocks of size $2\lceil \log_2 n \rceil$. For $i \in [n]$, let v^i be equal to 0 on all blocks but block $\Pi(i)$ and

on block Π it is $b(i)$. To see the backward implication of (*), let $A \in \mathcal{S}$ and $i \in [n]$ and assume $v^i \leq w^A$. A contains one element i' with $\Pi(i') = \Pi(i)$ and so we must have $b(i) \leq b(i')$ which implies $i = i'$.

Thus V and W can be generated in time $n^{O(1)}$. The partial function f will have domain $\{0,1\}^t$.

We need a few definitions. For $u \in \{0,1\}^t$:

- $D(u) = \{w : w \leq u\}$
- $\tau(u)$ denotes the DNF term $\prod_{i:u_i=0} \neg x_i$.

Note that $D(u)$ is exactly the set of satisfying assignments of $\tau(u)$. For a set U of vectors $D(U) = \bigcup_{u \in U} D(u)$. Note that, by (*), $V \subseteq D(W)$.

We define f by $f(x) = 1$ for $v \in V$, $f(x) = *$ if $x \in D(W) - V$, and $f(x) = 0$ otherwise. We claim that $\mathcal{S}$ has a cover of size m if and only if there is an m-term DNF consistent with f. Constructing the truth table for f takes time $2^{O(t)}$ which is polynomial in n.

Given a cover $\mathcal{C} \subseteq \mathcal{S}$ of size m, the m-term DNF whose terms are $\{\tau(w^C) : C \in \mathcal{C}\}$ is easily seen to be consistent with f.

Conversely, suppose ϕ is an m-term DNF consistent with f. For $\tau \in \phi$, let $u(\tau)$ be the maximal vector satisfying τ. Since ϕ is consistent with f, $u(\tau) \in D(W)$, so there must be a set $S(\tau) \in \mathcal{S}$ for which $u(\tau) \leq w^S$. We claim that $\{S(\tau) : \tau \in \phi\}$ is a cover of $\mathcal{S}$. Let $j \in [n]$ we must show that j is covered. By the consistency of ϕ, v^j is accepted by some term $\tau_j \in \phi$. This implies $v^j \leq u(\tau_j)$. Thus $v^j \leq S(\tau_j)$ which by (*) implies $j \in S(\tau_j)$.

□

3.2 Second phase of the *Min-DNF* NP-completeness reduction

As mentioned above, the second phase of our reduction is taken from Gimpel. We describe his reduction in detail here, with a correctness proof. We do this for two reasons – first, so that this paper will contain a self-contained simple proof that *Min-DNF* is NP-complete, and second, because we will build on the reduction later in order to prove non-approximability 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 $p(\vec{x})$ denotes the parity of $\vec{x}$ ($p(\vec{x})$ is 0 if the parity is even, and 1 if it is odd). Let $s = |f^{-1}(*)|$. The key observation is that every DNF for g requires s distinct terms to cover the inputs of the third type 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; 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.

Lemma 2. $\text{DNF-SIZE}(g) = \text{DNF-SIZE}(f) + s$

Proof. We first show that $\text{DNF-SIZE}(g) \leq \text{DNF-SIZE}(f) + s$. Suppose that 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 $\vec{x}y_1y_2$ 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 $\vec{x} \in f^{-1}(1) \cup f^{-1}(*)$) and $y_1 = y_2 = 1$

In all three cases we have $g(\vec{x}y_1y_2) = 1$, and thus ψ is consistent with g. The number of terms in ψ is $|f^{-1}(*)| + |\varphi| = \text{DNF-SIZE}(f) + s$.

We next show that $\text{DNF-SIZE}(g) \geq \text{DNF-SIZE}(f) + s$. 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.

Claim 3. *For every $\vec{x} \in f^{-1}(*)$, ψ contains the term*

$$t(\vec{x}) = \left(\bigwedge_{i:\vec{x}_i=1} x_i\right) \wedge \left(\bigwedge_{i:\vec{x}_i=0} \neg x_i\right) \wedge y_{2-p(\vec{x})}$$

Proof of claim 3. 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 10 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 $T = t(\vec{x})$. □

Claim 4. *There exist a subformula $\hat{\psi}$ of ψ and a DNF ψ' over the $\vec{x}$ variables that is consistent with f, such that*

$$\hat{\psi} = \bigvee_{T \in \psi'} (T \wedge y_1 \wedge y_2)$$

Proof of claim 4. Let $\hat{\psi}$ be the subformula of ψ consisting of those terms which 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 either 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 which is satisfied by $\vec{x}11$, and thus there is a corresponding term of ψ' which is 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. □

By Claims 3 and 4, ψ consists of the terms $t(\vec{x})$, for each $\vec{x} \in f^{-1}(*)$, together with the subformula $\hat{\psi}$. These components are pairwise disjoint, as every term of $\hat{\psi}$ contains y_1 and y_2, while each $t(\vec{x})$ contains only one of these. Since ψ' is consistent with f it follows that $\hat{\psi}$ contains at least DNF-SIZE(f) terms, and thus the size of ψ is at least DNF-SIZE$(f) + s$. □

Note that this second phase is, in fact, a reduction from *Min-DNF(*)* to *Min-DNF* .

Putting both phases together, we get a polynomial-time reduction from 3-Uniform Set Cover to *Min-DNF* , thus proving the NP-completeness of *Min-DNF* .

4 On the Approximability of *Min-DNF*

Although the above gives us a simple proof of the NP-completeness of *Min-DNF* , it does not give us inapproximability results for *Min-DNF* . There are two problems. First, the reduction begins with an instance of 3-Partite set cover, a problem that is easy to approximate to within a factor of $\ln 3$ [14, 20] (since the size of the largest subset is 3); we want to reduce from a problem that is difficult to approximate. Also, the second phase of the reduction is not approximation preserving.

We modify the first phase of the reduction so that instead of reducing from 3-Partite Set Cover, we reduce from r-Uniform Set Cover, for a non-constant value of r. This allows us to use known inapproximability results for r-Uniform Set Cover. However, because we are no longer reducing from a partite set cover problem, this requires a new way to construct the partial function f from the input set cover instance.

We modify the construction in the second phase of the reduction to make it approximation preserving.

4.1 Modified first phase

Our modified first phase uses the algorithm in the following lemma to convert an instance of r-Uniform Set Cover (we specify r later) into an instance of *Min-DNF(*)* .

Lemma 5. *There is an algorithm takes as input an r-uniform collection of subsets $\mathcal{S}$ over $[n]$, and produces the truth table of a partial boolean function f such that the minimum size of a cover of $[n]$ with $\mathcal{S}$ is equal to the minimum number of terms in a DNF consistent with f. The algorithm runs in time $(n|\mathcal{S}|)^{O(r)}$ and the number of variables of f is $O(r\log(n|\mathcal{S}|))$.*

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

As in the proof of Lemma 1, we produce two indexed sets of vectors $V = \{v^i : i \in [n]\}$ and $W = \{w^A : A \in \mathcal{S}\}$ of length t satisfying the property (*) that for all $A \in \mathcal{S}$ 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 $A \in \mathcal{S}$, 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 1, 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 $\mathcal{S}$.

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

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

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 $|P|e^{-t/3r} \leq |\mathcal{S}|ne^{-t/3r}$. Thus if $t \geq 3r(1 + \ln(|\mathcal{S}|n))$ 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., [4].) This is routine but technical so we provide only a sketch. Let $X(j, A)$ be the random variable which 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 $\mathbf{Exp}[X] \leq 1/2$. 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 $(|\mathcal{S}|n)^{O(1)})$. We can then use the method of conditional probabilities to fix these bits one at a time always choosing the value of the bit which 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 $(n|\mathcal{S}|)^{O(1)}$ with $t = O(r\log(n|\mathcal{S}|))$. Since the truth table has size 2^t, outputting it takes time $(n|\mathcal{S}|)^{O(r)}$. □

4.2 Modified Second Phase

We modify the reduction from *Min-DNF(*)* to *Min-DNF* to make it approximation preserving. Let f be a partial function over variables $x_1, \ldots, x_n$. Let $s = |f^{-1}(*)|$. We construct a new total function g' such that

$$\text{DNF-SIZE}(g') = s \cdot \text{DNF-SIZE}(f) + s = s \cdot (\text{DNF-SIZE}(f) + 1)$$

Let $t = 2 \log s$, 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

$$g'(\vec{x}\ y_1 y_2\ \vec{z}) = \begin{cases} 1, & \text{if } f(\vec{x}) = 1 \text{ and } y_1 = y_2 = 1 \text{ and } \vec{z} \in S \\ 1, & \text{if } f(\vec{x}) = * \text{ and } y_1 = y_2 = 1 \text{ and } \vec{z} \in \{0,1\}^t \\ 1, & \text{if } f(\vec{x}) = *, y_1 = p(\vec{x}), \text{ and } y_2 = \neg p(\vec{x}) \text{ and } \vec{z} \in \{0,1\}^t \\ 0, & \text{otherwise} \end{cases}$$

Lemma 6. $\text{DNF-SIZE}(g') = s \cdot \text{DNF-SIZE}(f) + s$

Proof. We first show that $\text{DNF-SIZE}(g') \leq s \cdot \text{DNF-SIZE}(f) + s$. Suppose that φ is a smallest 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 and every vector $\vec{z} \in S$, ψ contains the term

$$T \wedge y_1 \wedge y_2 \wedge \left(\bigwedge_{i:\vec{z}_i=1} z_i \right) \wedge \left(\bigwedge_{i:\vec{z}_i=0} \neg z_i \right)$$

These terms cover all inputs of the first type in the definition of g'. Finally, suppose that $\vec{x}y_1y_2\vec{z}$ 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 $g'(\vec{x}y_1y_2\vec{z}) = 1$, and thus ψ is consistent with g'. The number of terms in ψ is $|f^{-1}(*)| + |\varphi| \cdot |S| = s \cdot |\varphi| + s = s \cdot \text{DNF-SIZE}(f) + s$.

We next show that $\text{DNF-SIZE}(g') \geq s \cdot \text{DNF-SIZE}(f) + s$. Suppose that ψ is a smallest DNF for g'. The proof of the following claim is almost identical to the proof of Claim 3.

Claim 7. *For every $\vec{x} \in f^{-1}(*)$, ψ contains the term*

$$t(\vec{x}) = \left(\bigwedge_{i:\vec{x}_i=1} x_i \right) \wedge \left(\bigwedge_{i:\vec{x}_i=0} \neg x_i \right) \wedge y_{2-p(\vec{x})}$$

□

Claim 8. *For each $\vec{z} \in S$ there exist a subformula $\psi_{\vec{z}}$ of ψ, and a DNF $\psi'_{\vec{z}}$ over the $\vec{x}$ variables and consistent with f, such that*

$$\psi_{\vec{z}} = \bigvee_{T \in \psi'_{\vec{z}}} \left(T \wedge y_1 \wedge y_2 \wedge t(\vec{z})\right)$$

where

$$t(\vec{z}) = \left(\bigwedge_{i:\vec{z}_i=1} z_i\right) \wedge \left(\bigwedge_{i:\vec{z}_i=0} \neg z_i\right)$$

Proof of claim 8. Let $\vec{z} \in S$, and let $\psi_{\vec{z}}$ be the subformula of ψ consisting of those terms which are satisfied by $\vec{x}11\vec{z}$ for some $\vec{x} \in f^{-1}(1)$. Each term of $\psi_{\vec{z}}$ contains $y_1 \wedge y_2 \wedge t(\vec{z})$, as 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 which is satisfied by $\vec{x}11\vec{z}$, and thus there is a corresponding term of $\psi'_{\vec{z}}$ which 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.

By Claims 7 and 8, ψ consists of the terms $t(\vec{x})$, 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, as every term of $\psi_{\vec{z}}$ contains $t(\vec{z})$, and $t(\vec{x})$ contains no z-variables. Since $\psi'_{\vec{z}}$ is consistent with f it follows that $\psi_{\vec{z}}$ contains at least DNF-SIZE(f) terms, and thus the size of ψ is at least $s \cdot$ DNF-SIZE$(f) + s$. □

Combining the above two phases, we get a reduction from r-Uniform Set Cover to *Min-DNF* , which yields the following inapproximability result for *Min-DNF* .

Theorem 9. *For sufficiently large n and for any $r = \Omega(\log n)$, if NP is not in $BPTIME(n^{O(r)})$, then there is no polynomial-time $c(\log r)$ approximation algorithm for Min-DNF , for some constant c.*

Proof. Suppose that $\mathcal{A}$ is a polynomial-time $c' \log\log N$-approximation algorithm for *Min-DNF* , for some constant c'. We show how to use $\mathcal{A}$ to approximate r-uniform set cover.

Given an instance S of r-Uniform Set Cover over the ground elements $[n]$, by the reductions in the previous section we can obtain the truth table of a partial boolean function f over t variables, where $t = O(r \log(n|S|)$, such that the minimum size of a cover for S is equal to the minimum number of terms in a DNF consistent with f. This reduction takes time $poly(n)2^{O(t)}$. We can then obtain the truth table of a (total) boolean function g' over t' variables, where $t' = O(r \log(n|S|))$, such that the set cover instance has a cover of size q iff g' can be represented as a DNF of size $s(q+1)$, where s is the number of *'s in f.

Let N be the number of entries in the truth table of g'. Running $\mathcal{A}$ on the truth table of g' will produce a DNF φ of size at most $c's(q+1)\log\log N \leq c'sqd(\log r + \log\log(n|S|))$, for

some constant d. If $|S|$ is bounded by a fixed polynomial in n, then this quantity is at most $d'c'sq(\log r + \log\log n)$, for some constant d'. For $r = \Omega(\log n)$, by setting c' small enough, we get that this quantity is at most $sqc\log r$.

Thus (size of φ)/s is upper bounded by $qc\log r$. The value of s can be computed efficiently during the reduction, as can the size of φ. By computing (size of φ)/s, one thus obtains a value which is at most $c\log r$ larger than the size of the smallest cover for the original r-uniform set cover instance (and by the proofs of Lemmas 6 and 5, a set cover of size equal to that value can be generated in polynomial time from φ). It follows that r-uniform set cover can be approximated to within a factor of $c\log r$. But by [11], there is a constant c such that r-uniform set cover is not approximable to within a factor of $c\log r$ in time $n^{O(r)}$ unless $NP \in BPTIME(n^{O(r)})$. □

Note that when $r = \log n$ we have the following corollary:

Corollary 10. *There exists a constant c such that if NP is not in probabilistic quasipolynomial-time, then there is no polynomial-time, $c(\log\log N)$-approximation algorithm for Min-DNF . (Here, N is the input size for Min-DNF .)*

5 Improving the Inapproximability Factor

In this section we will improve our inapproximability factor to $c(\log N)^{\gamma}$, by exploiting the particular properties of the set cover instance obtained by the Lund/Yannakakis PCP-based reduction. We briefly review the Lund/Yannakakis reduction, as presented by Khot.

An instance of Label Cover (LC) is denoted by $\mathcal{L} = (G(V, W, E), L_1, L_2, \{\pi_{vw}\}_{(v,w)\in E})$ where: (1) $G(V, W, E)$ is a regular bipartite graph; (2) L_1 and L_2 are sets of labels where V and W get labels from L_1 and L_2 respectively; and (3) $\{\pi_{vw}\}_{(v,w)\in E}$ denote 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 $\pi_{vw}(l(w)) = l(v)$. Given an instance $\mathcal{L}$, the output should be a labelling which satisfies the maximum fraction, $OPT(\mathcal{L})$, of edges.

Theorem 11. *It is NP-hard to solve the following gap version of Label Cover. The input is an instance of Label Cover, $\mathcal{L} = (G(V, W, E), [7], [2], \{\pi_{vw}\}_{(v,w)\in E})$. The degree of each vertex in V is 5, and the degree of each vertex in W is 3. Also the size of V and W are n. (Note: The reduction is from Gap-Max3SAT, so V will correspond to the n underlying variables, and W will correspond to the m clauses.) The instance should be accepted if $OPT(\mathcal{L}) = 1$, and the instance should be rejected if $OPT(\mathcal{L})$ is at most c' for some fixed constant $c' < 1$.*

Using Raz's parallel repetition theorem, we can amplify the gap to get the following NP-hardness result for Label Cover.

Theorem 12. *For all k, it is NP-hard to solve the following gap version of Label Cover. The input is an instance of Label Cover $\mathcal{L}' = (G(V', W', E'), [7^k], [2^k], \{\pi'_{vw}\}_{(v,w)\in E})$, where: (1) $V' = V^k$ (so $|V'| = n^k$) and $W' = W^k$; (2) $L_1 = L_1^k$ and $L_2 = L_2^k$; (3) $(v', w') \in E'$ if and only if $(v_i, w_i) \in E$ for all $1 \le i \le k$, where $v' = (v_1, \ldots, v_k)$ and $w' = (w_1, \ldots, w_k)$; and (4)*

$\pi'_{v'w'}(b_1, \ldots, b_k) = \pi_{v_1w_1}(b_1), \ldots, \pi_{v_kw_k}(b_k))$. *The input should be accepted if* $OPT(\mathcal{L}') = 1$, *and the input should be rejected if* $OPT(\mathcal{L}')$ *is at most* $2^{-\gamma k}$ *for some absolute constant* $\gamma > 0$.

Note that $|V'|$ *and* $|W'|$ *are* $n^{O(k)}$, *where* n *is the size of the SAT instance. Also, the degree of each node in* V' *is* 5^k *and the degree of each node in* W' *is* 3^k. *Also, the number of possible labels on the left side is* 7^k *and on the right side is* 2^k. *We will choose* $k = \log\log n$.

Definition 13. *A Partition System* $\mathcal{P}(\mathcal{U}, m, h, t)$ *consists of: (1) a universe* $\mathcal{U}$ *of* m *elements; (2)* t *pairs of disjoint sets* $(A_1, \overline{A_1}), \ldots (A_t, \overline{A_t})$, *with* $A_i \subset [m]$. *Furthermore it has the property that no collection of* h *sets, with at most one set from any pair, can cover the whole universe.*

Lemma 14. *Given* h, t, *there is a partition system with* $m = O(2^h h log t)$.

We now review the reduction from Label Cover instance, $\mathcal{L}'$ to Set Cover, $SC_{\mathcal{L}'}$. It has been proven that if $OPT(\mathcal{L}') = 1$ then our set cover instance has a cover of size $|V'| + |W'|$. On the other hand, if $OPT(\mathcal{L}') \leq 2^{-\gamma k}$, then our set cover instance requires a cover larger than $\rho h|W'|$ where h is a parameter in our construction and ρ is an absolute constant. Thus we get an $\Omega(h)$ inapproximability result for set cover.

We assume without loss of generality that $|V'| = |W'|$. (Make copies of every vertex in V' so that this is the case. This will preserve the value of $OPT(\mathcal{L}')$.) First, the universe $\mathcal{U}$ of $SC_{\mathcal{L}'}$ is as follows. For each edge $e = (v, w)$ we associate a subuniverse of 2^{h+1} elements, $\mathcal{U}_{vw} = \{(e, i), i \leq 2^{h+1}\}$. The entire universe is the disjoint union of these $|E|$ subuniverses. Associated with each edge (v, w) will be a partition system over $\mathcal{U}_{vw}$, with one partition associated with each of the possible labels. Call the partitions over $\mathcal{U}_{vw}$: $(A_1^{vw}, \overline{A}_1^{vw}), \ldots, (A_l^{vw}, \overline{A}_l^{vw})$. Thus the size of the entire universe is $n^{O(k)}2^{O(h)}$.

We now define the sets. The sets will be the union of two collections of sets, $S(v, i)$, for each vertex $v \in V$ and each possible label $i \in L_1$, and sets $S(w, j)$, for each $w \in W$ and label $j \in L_2$. In particular,

$$S(v, i) = \cup_{w:(v,w)\in E} A_i^{vw}.$$

$$S(w, j) = \cup_{w:(v,w)\in E} \overline{A}^{vw}_{\pi_{vw}(j)}.$$

The following two claims are implicit in Lund/Yannakakis as well as in later improvements.

Claim 15. *If* $OPT(\mathcal{L}) = 1$ *then* $SC_{\mathcal{L}}$ *has a cover of size* $|V| + |W| = n^{O(k)}$.

Claim 16. *Setting* $h = O(2^{\gamma k/2})$, *if* $OPT(\mathcal{L}) \leq 2^{-\gamma k}$, *then* $SC_{\mathcal{L}}$ *requires a cover of size* $h|W|/8$.

By setting $k = O(\log\log n)$, the above claims imply that we cannot approximate set cover to within an $O(\log n)$ factor unless NP is contained in quasipolynomial time.

It is left to show how to reduce instances of set cover of the above form, $SC_{\mathcal{L}}$ to *Min-DNF(*)* . We assume that $(\mathcal{S}, \mathcal{U})$ has the property that there are no two elements $(e, i), (e', i') \in \mathcal{U}$ such that $(e, i) \in S \iff (e', i') \in S$ for all $S \in \mathcal{S}$. If there exist such elements, they can easily be identified and all but one of them removed without changing the size of an optimum set cover. We

now describe how to encode $(\mathcal{S}, \mathcal{U})$ as an instance of MinDNF*. By the results of section 3.1 it suffices to define vectors

$$\{u_{e,i} \mid (e,i) \in \mathcal{U}\}$$
$$\{t_{v,a} \mid v \in V,\ a \in L_1\}$$
$$\{t_{w,b} \mid w \in W,\ b \in L_2\}$$

such that the following conditions hold.

$$u_{e,i} \leq t_{v,a} \iff (e,i) \in S(v,a) \qquad \forall (e,i) \in \mathcal{U},\ v \in V, a \in L_1$$
$$u_{e,i} \leq t_{w,b} \iff (e,i) \in S(w,b) \qquad \forall (e,i) \in \mathcal{U},\ w \in W, b \in L_2$$

Let $m \in O(\log |V|)$ be such that $\binom{m}{m/2} \geq |V|, |W|$. Our function will have variables

$$\{x_1, \ldots, x_m\} \cup \{x'_1, \ldots, x'_m\} \cup \{y_a \mid a \in L_1\} \cup \{y'_b \mid b \in L_2\}$$

Thus the number of variables is

$$O(\log |V| + |L_1| + L_2|) = O(k \log n + 7^k + 2^k)$$

We assign to each $v \in V$ a unique set $S_v \subseteq \{1, \ldots, m\}$ of size $m/2$; and similarly each $w \in W$ is assigned a unique set $S_w \subseteq \{1, \ldots, m\}$ of size $m/2$. 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

$$S(v,a_1), \ldots, S(v,a_k)$$
$$S(w,b_1), \ldots, S(w,b_\ell)$$

be all of the sets in $\mathcal{S}$ containing (e,i). Then $u_{e,i}$ has zeroes in the positions corresponding to the following variables.

- Variables x_i, where $i \in S_v$
- Variables x'_i, where $i \in S_w$
- Variables y_{a_i}, where $1 \leq i \leq k$
- Variables y'_{b_i}, where $1 \leq i \leq \ell$

The remaining bits of $u_{e,i}$ are ones.

Claim 17. *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.*

$$u_{e,i} \leq t_{v,a} \iff (e,i) \in S(v,a)$$
$$u_{e,i} \leq t_{w,b} \iff (e,i) \in S(w,b)$$

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 $w \in W$. Now 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 MinDNF* on $O(k \log n + 7^k + 2^k)$ variables whose optimum is equal to the optimum for the instance $(\mathcal{S}, \mathcal{U})$ of SET-COVER. We have therefore nearly proven the following theorem.

Theorem 18. *If $NP \not\subseteq QP$ then there exists an absolute constant $0 < \delta < 1$ such that there is no polynomial-time, $(\log N)^\delta$-approximation algorithm for MinDNF*, where N is the size of the truth table.*

Proof. Let f be the partial function specified by our reduction. The reduction of Lund and Yannakakis, together with the results of section 3.1, implies that our MinDNF* instance has the following properties:

- If $\text{OPT}(G, L_1, L_2, \Pi) = 1$ then $\text{DNF-SIZE}(f) \leq 2 \cdot |V|$.
- If $\text{OPT}(G, L_1, L_2, \Pi) \leq 2^{-\gamma k}$ then $\text{DNF-SIZE}(f) \geq h \cdot |V|/8$

where $h = O(2^{k\gamma/2})$. Let us take $k = \log\log n$, and thus $h = O((\log n)^{\gamma/2})$. The number of variables of f is

$$\begin{aligned} O(k \log n + 7^k + 2^k) &= O(\log n \log\log n + (\log n)^{\log 7} + \log n) \\ &= O((\log n)^{\log 7}) \\ &= \log N \end{aligned}$$

Thus a polynomial-time, $(h/16)$-approximation for MinDNF* would imply a quasi-polynomial-time solution to the LABEL-COVER instance. In terms of N, the size of the truth table, this lower bound translates to

$$h/16 = c(\log n)^{\gamma/2} = \Omega\left((\log N)^{\gamma/(2\log 7)}\right)$$

The theorem follows by taking $\delta = \gamma/(2 \log 7)$. □

The results of section 3.2, together with theorem 18, yield the following hardness result for MinDNF.

Theorem 19. *If $NP \not\subseteq QP$ then there exists an absolute constant $0 < \epsilon < 1$ such that there is no polynomial-time, $(\log N)^\epsilon$-approximation algorithm for MinDNF, where N is the size of the truth table.*

6 Fixed Parameter Complexity of *Min-DNF*

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 [13]. (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 [9].) Thus, *Min-DNF* is *fixed parameter tractable*. 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 20. *If there exists some constant $\epsilon > 0$ such that NP is not contained in $DTIME(2^{O(n^\epsilon)})$, 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 of size $s = O(n^b)$ for some constant $b > 1$. 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 $m - n$ new dummy variables. Suppose *Min-DNF* is solvable in polynomial time when $k = n^c$. Then the padded instance of *Min-DNF* can be solved in time polynomial in 2^s, and Π can be solved in time $2^{O(n^{b*\frac{1}{c}})}$. For $c > \frac{b}{\epsilon}$, this is less than $2^{O(n^\epsilon)}$. Contradiction. □

Our interest in the parametrized version of *Min-DNF* is motivated in part by connections to computational learning theory. One of the major problems in learning theory is to determine whether DNF formulas can be learned in polynomial time. Since this problem is difficult and remains open for many learning models, there has been a significant amount of research on learning k-term DNF formulas for small values of k (see e.g. [5, 7, 19, 13]). The connection between learning k-term DNF formulas and *Min-DNF* for k is strongest for "proper" learning models: in such models, any hypotheses used in learning the class of k-term DNF formulas must themselves be k-term DNF formulas.

Pitt and Valiant showed that in the proper PAC model, unless RP=NP you cannot properly learn k-term DNF formulas in polynomial time (for constant k). Their proof was based on showing that finding a k-term DNF consistent with a sample is NP-hard. This consistency problem is related to the *Min-DNF(*)* problem, the difference being that in the consistency problem the sample consists only of the elements in $f^{-1}(1) \bigcup f^{-1}(0)$, for formula f, whereas in *Min-DNF(*)* the whole truth table is given as input. Their NP-hardness proof can be easily extended to k-term DNF for larger values of k, yielding non-learnability results for larger values of k in the PAC model. We note that our reduction to *Min-DNF(*)* in fact implies that the k-term DNF consistency problem 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 cannot be accomplished in polynomial time unless RP=NP.

Pillaipakkamnatt and Raghavan showed that for some $\epsilon < 1$ and some $c > 1$, $\log^c n$-term DNF cannot be learned in the membership and proper equivalence query model unless $NP \subseteq DTIME(2^{O(n^\epsilon)})$. Subsequently, Hellerstein and Raghavan proved that $\Omega(\log^{3+\epsilon} n)$-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. (It can be improved to $\Omega(\log^{2+\epsilon}$.) 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 [7]. A polynomial-time algorithm for learning $\log n$-term DNF formulas in this model would imply a polynomial-time algorithm for *Min-DNF* for $k = n$ [13]. 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 $k = n^c$. Thus the result of Pillaipakkamnatt and Raghavan mentioned above can also be derived from Corollary 20.

7 Hardness of *Min-AC*^0_d

In [2] it was shown that neither *Min-Circuit* nor *Min-NC*^{1} can be approximated to within a factor of $n^{1-\epsilon}$ in polynomial time unless Blum Integers can be factored efficiently. In this section, we strengthen their result to hold for *Min-AC*^{0} as well. More precisely, let *Min-AC*^0_d be the problem of estimating the size of depth d AC^0 circuit size. For large enough values of d, *Min-AC*^0_d is difficult to estimate, unless factoring Blum integers can be performed in time 2^{n^δ}, where the value of d depends on the value of δ. We have not yet computed the relationship between d and δ, but we anticipate that this yields a meaningful inapproximability result for d as small as 10.

Lemma 21. *For every language $\mathcal{L}$ in NL, and for every ϵ, there exists a d such that there are AC^0_d circuits of size 2^{n^ϵ} that recognize $\mathcal{L}$.*

Proof. (Proof sketch) 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 $\sqrt{m}$ connecting them. This gives rise to a DNF formula of size roughly $2^{\sqrt{m}}$. Iterating this idea gives depth d AC^0 circuits of size 2^{n^ϵ}. This is basically a strengthening of Nepomnascii's Theorem [3, 23] □

Definition 22. *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 23. *For every $\epsilon > 0$ there is a depth d such that Blum Integer Factorization is in BPTime$(2^{n^\epsilon})^B$, where B is any function that approximates Min-AC^0_d to within a factor of $n^{1-\delta}$ (for any $\delta > 0$).*

Proof. We follow the proof given in [2]. In [22] a pseudo-random function emsemble $\{f_{N,r}(x) : \{0,1\}^n \to \{0,1\}\}_{N,r}$ 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 $\mathcal{M}$ running in time $t(n)$ with oracle access to $f_{N,r}$ of query length n, there exists a probabilistic Turing machine $\mathcal{A}$ running in time $t(n)n^{O(1)}$ such that for every $N = PQ$, $|N| = 2n$, if $|Pr[\mathcal{M}^{f_{N,r}}(N) = 1] - Pr[\mathcal{M}^{R_n}(N) = 1]| > 1/2$, where R_n is a uniformly distributed random function ensemble, and the probability is taken over random r, and random bits of $\mathcal{M}$ then $Pr[\mathcal{A}(N) \in \{P, Q\}] > 1/2$. In other words, if $\mathcal{M}$ can distinguish the pseudorandom function ensemble from a truly random function "efficiently", then Blum Integers can be factored in "efficiently" on a probabilistic machine.

Now let $f_{N,r}$ be computable by a TC^0 circuit of size $n^{c'}$. 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^ϵ}. For all large enough n, all $2n$-bit integers n and all $4n^2 + 2n$-bit strings r, consider the function h 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 21, there is some d such that this function has depth d AC^0 circuits of size 2^{n^γ} for some $\gamma < \epsilon$.

Now consider the following oracle Turing machine $\mathcal{M}$ with with oracle access to g and $\mathcal{B}$: On input N, $\mathcal{M}$ 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 $y_1, \ldots, y_m$. Then $\mathcal{M}$ accepts N if and only if $\mathcal{B}$, on input h, with $k = m^{1-\delta/2}$ rejects. (That is, $\mathcal{M}$ accepts if and only if the oracle $\mathcal{B}$ is estimating that h requires large AC^0_d circuits.) We are assuming that $\mathcal{B}$ approximates *Min-AC*^0_d within a factor of $m^{1-\delta}$, i.e., that $1 \leq \mathcal{B}(h)/$*Min-AC*$^0_d(h) \leq m^{1-\delta}$. Now for sufficiently large n, if $g \in \{f_{N,r}(x)\}_{N,r}$ then $\mathcal{M}$ will always reject, since as argued above, in this case h has AC^0_d circuits of size at most 2^{n^γ}, and $2^{n^\gamma} m^{1-\delta} < m^{1-\delta/2}$. On the other hand, if g is taken uniformly at random from $\mathcal{R}_n$, then $y_1 \ldots y_m$ is a random string and since almost all functions require AC^0_d circuits of size $m/\log m > m^{1-\delta/2}$, on all such inputs it follows that $\mathcal{M}$ will accept, and hence $|Pr[\mathcal{M}^{f_{N,r}(x)}(N) = 1] - Pr[\mathcal{M}^{R_n}(N) = 1]| > 1/2$, for sufficiently large n. Thus, $\mathcal{M}$ 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.

□

As a corollary to the above theorem we have:

Theorem 24. *For all $\epsilon > 0$ there exists a d such that for all $\delta > 0$,* Min-AC^0_d *cannot be approximated to within a factor $n^{1-\delta}$ in BPP unless Blum Integer Factorization is in BPTime(2^{n^ϵ}).*

8 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 [8, 13, 1], and results on learning circuits [6] yield positive results for approximating circuit minimization (cf. [28]). 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.

NP-hardness of proper PAC learning DNF, and *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. This 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.

Finally, an open problem is to close the approximability gap for *Min-DNF* .

9 Acknowledgements

The authors gratefully acknowledge Uri Feige and Subhash Khot for helpful conversations on the inapproximability of partite restrictions of set cover.

References

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

[2] E. Allender, M. Koucky, D. Ronneberger, and S. Roy. Derandomization and distinguishing complexity. In *18th IEEE Conference on Computational Complexity*, pages 209–220, 2003.

[3] E. Allender, M. Koucky, D. Ronneburger, and S. Roy. Time-space tradeoffs in the counting hierarchy. In *16th IEEE Conference on Computational Complexity*, pages 295–302, 2001.

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

[5] U. Berggren. Linear time deterministic learning of k-term DNF. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 37–40. ACM Press, 1993.

[6] N. Bshouty, R. Cleve, R. Gavalda, S. Kannan, and C. Tamon. *Journal of Computer and System Sciences*, 3(52):421–433, 1996.

[7] N. H. Bshouty. Simple learning algorithms using divide and conquer. *Computational Complexity*, 6:174–194, 1997.

[8] 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 (COLT'02)*, volume LNAI 2375, pages 287–302. Springer Verlag, 2002.

[9] A. Chandra and G. Markowsky. On the number of prime implicants. *Discrete Mathematics*, 24:7–11, 1978.

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

[11] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, July 1998.

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

[13] L. Hellerstein and V. Raghavan. Exact learning of DNF formulas using DNF hypothesis. *J. of Computer and System Sciences*, 70(4):435–470, 2005.

[14] D. S. Johnson. Approximation algorithms for combinatorial problems. J. Comput. System Sci., 9:256–278, 1974.

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

[16] V. Kabanets and J. L. Cai. Circuit minimization problem. In *STOC*, pages 73–79, 2000.

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

[18] K. Krajíček and P. Pudlák. Some consequences of cryptographic conjectures for S_2^1 and EF. In D. Leivant, editor, *Logic and Computational Complexity: international workshop, LCC '94*, volume 960 of *Lecture Notes in Computer Science*, pages 210–220. Springer-Verlag, 1995.

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

[20] On the ratio of optimal integral and fractional covers. *Discrete Math*, 13:383–390, 1975.

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

[22] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudorandom functions. In *IEEE FOCS*, pages 458–467, 1997.

[23] V. A. Nepomnjascii. Rudimentary predicates and turing calculations. *Math. Dokl*, 11:1462–1465, 1970.

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

[25] A. A. Razborov and S. Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, August 1997.

[26] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[27] C. Umans. The minimum equivalent DNF problem and shortest implicants. In *IEEE Symposium on Foundations of Computer Science*, pages 556–563, 1998.

[28] C. Umans. Hardness of approximating Σ_2^p minimization problems. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 465–474, 1999.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'