Yale University
Department of Computer Science

One Help Bit Doesn't

Richard Beigel
Yale University and U. Maryland

Tirza Hirst
Bar Ilan University

YALEU/DCS/TR-1118
December 1996

One Help Bit Doesn't

Richard Beigel*
Yale University and U. Maryland

Tirza Hirst†
Bar Ilan University

Abstract

We show that nontrivial frequency computation is impossible in the decision-tree model, and therefore k decision forests cannot nontrivially cover k functions. This implies that $\lfloor \log_2 k \rfloor$ help bits do not help you to evaluate a function on k inputs, which is tight. In particular, 1 help bit does not help you to compute anything, answering a question of Nisan, Rudich, and Saks (FOCS '94).

1. Background and Definitions

Think of your favorite computational model. Let f be a Boolean function and let f's complexity be c in your model. We look at several variants on computing f.

Help Bits. Suppose that you want to evaluate f on two inputs x and y, and you are given for free one "help bit," i.e., the value of an arbitrary Boolean function H evaluated at the single point (x, y). Is it possible to design the help-bit function H so that using the help bit you can then determine the values of $f(x)$ and $f(b)$ via computations whose complexity is less than c each? What if you want to evaluate f on k inputs and you are allowed h help bits?

Covers. Suppose that you want to evaluate f on two inputs x and y and give two pairs of answers (a_x^1, a_y^1) and (a_x^2, a_y^2) for (x, y), but only one pair has to be correct. Does complexity less than c ever suffice? What if you want to evaluate f on k inputs and you are allowed to give m answer tuples?

*On sabbatical from Yale 1996–97. Address: Dept. of Computer Science, University of Maryland at College Park, College Park, MD 20742-3251, USA. Supported in part by the National Science Foundation under grants CCR-8958528 and CCR-9415410 and by NASA under grant NAG 52895. Email: `beigel@cs.umd.edu`.

†Department of Mathematics and Computer Science, Bar-Ilan University, 52900 Ramat-Gan, Israel. Tel: (972-3)531-8407. Research performed in part while a postdoc at Yale with support from the Wolfson Scholarship Program. Email: `tirza@cs.biu.ac.il`.

Frequency. Suppose that you want to evaluate f on two inputs x and y. You give answers a_x and a_y for each input, but only one of your two answers has to be right. Does this problem have a solution whose complexity is less than c? What if you want to evaluate f on m inputs, but only k of your answers have to be right?

Probabilistic. Suppose that you want to evaluate f on one input. You are allowed to give m answers but at least k of them have to be correct.

We will give some formal definitions for the concepts described informally above. In this paper we will be concerned only with the decision-tree model of computation.

Definition 1 (Decision Tree).

- A *decision tree over a variable set* X is a triple (T, p, a) where T is a rooted ordered binary tree, p is a mapping from the internal nodes of T to the set X, and a is a mapping from the leaves of T to $\{0, 1\}$. (p is said to label the nodes of T.)
- A decision tree T over X computes a boolean function f as follows: If α is any assignment in $\{0,1\}^X$, the *computation* of T on α is the unique path $v_0, v_1, \ldots, v_s$ from the root v_0 of T to some leaf v_s. Inductively, v_{i+1} for $i \geq 0$ is defined to be the $\alpha(p(v_i))$th child of v_i. The output of the computation is $a(v_s)$.
- The *depth of a decision tree* T is the length of the longest path in T.
- DEPTH (d) is the class of boolean functions computable by decision trees having depth d or less.

Definition 2 (Decision Forest).

- A *decision forest* over a variable set X is a tuple of decision trees over X (note that order matters).
- The forest $F = (T_1, \ldots, T_k)$ computes the function

$$F(\alpha_1, \ldots, \alpha_k) = (T_1(\alpha_1), \ldots, T_k(\alpha_k)).$$

- The *depth* of the forest $(T_1, \ldots, T_k)$ is the maximum of the depths of $T_1, \ldots, T_k$.

Nisan et al. [18] defined a notion of cover that is analogous to classical definitions in polynomial-time and recursion-theoretic models of bounded queries — Beigel's [3, 4] "computability by a set of m polynomial-time computable functions" or by "m partial recursive functions" and Cai and Hemachandra's [10] "m-enumerability."

Definition 3 (Cover).

- A *family* of decision forests over X is a set of decision forests over X.

- The *depth* of a family of decision forests $\{F_1, \ldots, F_k\}$ is the maximum of the depths of $F_1, \ldots, F_k$.
- The *size* of a family of forests $\mathcal{F}$ is the number of forests in $\mathcal{F}$.
- A family $\mathcal{F}$ of forests *covers* a function g if

$$(\forall \alpha)(\exists F \in \mathcal{F})[F(\alpha) = g(\alpha)].$$

  If $\mathcal{F}$ covers g, we call $\mathcal{F}$ a *cover* for g.

Definition 4 (Pure). Let X be the disjoint union of $X_1, \ldots, X_k$ and, for $1 \leq j \leq k$, let f_j be a function over X_j. Let

$$\mathcal{F} = \{(T_{11}, \ldots, T_{1k}), \ldots, (T_{m1}, \ldots, T_{mk})\}$$

be a cover for $(f_1, \ldots, f_k)$.

- For $1 \leq i \leq m$, X_j is called the set of variables *internal* to T_{ij}.
- For $1 \leq i \leq m$, $X - X_j$ is called the set of variables *external* to T_{ij}.
- The tree T_{ij} is called *pure* if its nodes are labeled only with internal variables, *impure* otherwise.
- A forest is *pure* if it consists only of pure trees, *impure* otherwise.
- A cover is *pure* if it consists only of pure forests, *impure* otherwise.

Intuitively it would seem optimal to label T_i only with variables upon which f_i depends, so impure covers "ought to" be at least as big and deep as pure covers. However, Nisan et al. [18] state that this question is open. In our opinion, this is the most important open problem concerning bounded queries in the decision-tree model.

Help bits are Cai's name [9] for queries to an arbitrarily powerful set (see also [1, 18]). Covers are a nice refinement of help bits, because in every model of computation where bounded queries have been studied [3, 4, 1, 18] — including the decision-tree and pure–decision-tree models — the following is known:

Proposition 5. *g is computable with h help bits iff g has a cover of size 2^h.*

Proof sketch: For the forward direction you have one element in the cover for each setting of the h help bits. Conversely, for each input the h help bits give an index for the element of the cover that is correct on that input. △

We are mainly interested in covers for a tuple g consisting of k isomorphic copies of the same function (over disjoint sets of variables).

Definition 6. Let f be a function from $\{0,1\}^n$ to $\{0,1\}$. Define

$$f^{[k]}(\vec{x}_1, \ldots, \vec{x}_k) = (f(\vec{x}_1), \ldots, f(\vec{x}_k)).$$

Definition 7. f is (pure) *m for k in depth d* if there exists a (pure) size-m, depth-d cover for $f^{[k]}$. (When d is understood, we will say simply that f is *m for k*.) Let $\mathrm{Cover}_d\,(m,k)$ denote the class of Boolean functions that are m for k in depth d. Let $\mathrm{PCover}_d\,(m,k)$ denote the class of Boolean functions that are pure m for k in depth d.

By brute-force constructions, it is clear that (1) every Boolean function is pure 2^k for k; and (2) if f is (pure) m for k then f is (pure) m^i for ki for all $i \geq 1$. Nisan et al. [18] asked

- Is $\mathrm{Cover}_d\,(m,k) = \mathrm{PCover}_d\,(m,k)$ for all d, m, k?
- Is the simplest cover class nontrivial, i.e., is there a function in $\mathrm{Cover}_d\,(2,2) - \mathrm{DEPTH}\,(d)$ for some d?

They made some progress on their second question, showing that $\mathrm{PCover}_d\,(2,2) = \mathrm{DEPTH}\,(d)$ for all d.

We answer their second question completely:

$$(\forall d)(\forall k)[\mathrm{Cover}_d\,(k,k) = \mathrm{PCover}_d\,(k,k) = \mathrm{DEPTH}\,(d)].$$

This is analogous to the result in recursion theory that f is k for k in a finite number of steps iff k is recursive, but the proof is quite different.

Nisan, Rudich, and Saks [18] determined the asymptotic number of help bits needed in order for a depth-d decision forest to evaluate a Boolean function f on k inputs. Still, nothing was known about fixed k. We prove a gap between 0 and $\lfloor \log_2 k \rfloor + 1$ help bits: either f can be evaluated in depth d without any help bits, or else $\lfloor \log_2 k \rfloor + 1$ help bits are required in order to evaluate k instances in depth d. In particular, one help bit is as good as none. There exist functions for which our gap is the best possible.

In recursion theory, polynomial-time computation, and circuit complexity, "$k+1$ for k" is qualitatively different from "k for k." Hence we would expect $\mathrm{Cover}_d\,(k+1,k)$ to behave more interestingly than $\mathrm{Cover}_d\,(k,k)$, and it does:

$$(\forall k)(\forall d \geq k-1)[\mathrm{DEPTH}\,(d) \subset \mathrm{PCover}_d\,(k+1,k) \subseteq \mathrm{DEPTH}\,(3d)].$$

We prove the first nontrivial equality between pure cover classes:

$$(\forall d)[\mathrm{PCover}_d\,(3,2) = \mathrm{PCover}_d\,(5,3)].$$

No other cover classes or pure cover classes are known to be equal to each other, unless they are equal to $\mathrm{DEPTH}\,(d)$.

In frequency computation, you are given m problem instances and you try to answer at least k of them correctly. Frequency computation was defined by Rose [19] and studied further by Myhill (see [17, p. 393]), Trakhtenbrot [20], and others [13, 11, 15]. Connections between frequency computation and bounded queries were studied in [14, 12, 8, 7, 6].

Definition 8 (Frequency Computation). f is *k in m* in depth d if there exist m depth-d decision trees $T_1, \ldots, T_m$ such that

$$(\forall \alpha_1, \ldots, \alpha_m)[|\{i : T_i(\alpha_i) = f(\alpha_i)\}| \geq k].$$

Let $\mathrm{Freq}_d(k, m)$ denote the class of Boolean functions that are k in m in depth d.

In circuit and Turing-machine computation models, it is easy to see that f is 1 in m iff f is $2^m - 1$ for m [1, 8]. However, this result does not carry over to decision trees. In fact, we will prove

$$(\forall m \geq 1)\left[\mathrm{Freq}_d(1, m) = \mathrm{DEPTH}(d) \subset \mathrm{PCover}_d(2^m - 1, m)\right].$$

Thus frequency computation in the decision-tree model is completely different from frequency computation in the circuit and Turing-machine models. The impossibility of frequency computation implies our main result that

$$(\forall m \geq 1)\left[\mathrm{Cover}_d(m, m) = \mathrm{DEPTH}(d)\right].$$

We show that a function has nontrivial pure covers iff it is reducible to MAJ_{2n+1} for some n. Thus it would seem that understanding MAJ_{2n+1} might be the key to understanding the pure cover classes. We show how to construct nearly optimal covers for $\mathrm{MAJ}_{2n+1}^{[k]}$ (the existence of slightly better covers was shown previously by a probabilistic argument [18]). By reducing to MAJ_{2n+1} we construct nontrivial pure covers for all functions that have them.

In probabilistic computation, you are given one problem instance and m chances to solve it, and you try to get the answer right at least k times.

Definition 9 (Probabilistic Computation). f is *k out of m* (in depth d) if there exist m depth-d decision trees such that for each input α, at least k trees yield a correct answer for $f(\alpha)$. Let $\mathrm{Prob}_d(k, m)$ denote the class of Boolean functions that are k out of m in depth d.

It might seem that f is k out of m iff depth-d decision trees can compute f with probability k/m. However it is unknown whether $\mathrm{Prob}_d(k, m) = \mathrm{Prob}_d(2k, 2m)$.

We will prove that $\mathrm{Prob}_d(2, 3) = \mathrm{PCover}_d(3, 2)$, which implies that the function MAJ_3 is complete for $\mathrm{PCover}_d(3, 2)$ under depth-d-tree reductions. It is also interesting that MAJ_3 is equivalent to OR_2, in the sense that

$$\mathrm{MAJ}_3 \in \mathrm{PCover}_1(m, k) \iff \mathrm{OR}_2 \in \mathrm{PCover}_1(m, k).$$

You might think that this follows from reductions between MAJ_3 and OR_2. This is not the case. While OR_2 is obviously depth-1-tree reducible to MAJ_3, MAJ_3 is not depth-1-tree reducible to OR_2.

2. Examples and Notation for Covers

We will use the following notation when presenting a family of forests $\mathcal{F} = \{F_1, \ldots, F_m\}$ where $F_i = (T_1^i, \ldots, T_k^i)$:

$$\begin{array}{c} \boxed{T_1^1 \ \cdots \ T_k^1} \\ \vdots \\ \boxed{T_1^m \ \cdots \ T_k^m} \end{array}$$

Often we write a function in place of a tree if it is obvious how to compute the function with a decision tree.

The following example was first suggested to Nisan et al. [18] by Manuel Blum:

Example 10. Define the Boolean function

$$\mathrm{MAJ}_3(x, y, z) = \begin{cases} 1 & \text{if } x + y + z \geq 2 \\ 0 & \text{otherwise} \end{cases}$$

Let $X_1 = \{x_1, y_1, z_1\}$, $X_2 = \{x_2, y_2, z_2\}$, and $X = X_1 \cup X_2$. MAJ_3 is pure 3 for 2 in depth 1 via the following cover:

x_1	x_2
y_1	y_2
z_1	z_2

This is easily seen to be a cover because at most one of the variables x_i, y_i, z_i can be unequal to $\mathrm{MAJ}_3(x_i, y_i, z_i)$. As a convention, *when discussing pure covers we will drop the subscript that distinguishes variables from different problem instances.* Accordingly, we represent the cover above as follows:

x	x
y	y
z	z

Example 11. Let $\mathrm{OR}_n(x_1, \ldots, x_n) = x_1 \vee \cdots \vee x_n$. OR_2 is pure 5 for 3 in depth 1 via the following cover:

x	x	x
1	y	y
y	1	y
y	y	1
1	1	1

Consider an assignment to three instances of OR_2. If all three answers are 1, the fifth forest covers the assignment. Notice that if $\mathrm{OR}(x, y) = 0$ then $x = y$, so it doesn't matter which variable a decision tree checks. Thus, if all three answers are 0 the first forest covers the assignment, and if exactly two answers are 0, one of the middle three forests covers it.

The only case remaining is when exactly two answers are 1. If $x = 1$ in two instances, then the first forest covers the assignment. If $y = 1$ in one instance and $x = 1$ in another, then choose the forest that has 1 in the column where $x = 1$ and has y in the other two columns. Otherwise $y = 1$ in two instances; again one of the middle three forests covers the assignment.

Example 12. If $k > n(n-1)/2$ then OR_n is pure $2^k - 1$ for k in depth 1 via the following cover:

$$
\begin{array}{|cccccc|}
\hline
1 & & \ldots & & 1 & x_1 \\
\hline
1 & & \ldots & 1 & x_1 & 1 \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
x_1 & 1 & & \ldots & & 1 \\
\hline
\hline
1 & & \ldots & 1 & x_2 & x_2 \\
\hline
1 & \ldots & 1 & x_2 & x_2 & 1 \\
\hline
1 & \ldots & 1 & x_2 & 1 & x_2 \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
x_2 & x_2 & 1 & & \ldots & 1 \\
\hline
\multicolumn{6}{c}{\vdots} \\
\hline
x_n & & & \ldots & & x_n \\
\hline
\end{array}
$$

Let $\alpha = (\alpha_1, \ldots, \alpha_k)$.

Case 1: there exists i such that $\mathrm{OR}_n(\alpha_i) = 0$. Let $S = \{i : \mathrm{OR}_n(\alpha_i) = 0\}$. Let $T_i = x_{|S|}$ if $i \in S$, 1 otherwise. Then $(T_1, \ldots, T_k)$ is an element of our proposed cover, and $(T_1, \ldots, T_k)(\alpha) = \mathrm{OR}_n^{[k]}(\alpha)$

Case 2: for all i, $\mathrm{OR}_n(\alpha_i) = 1$. We have

$$
\begin{aligned}
\sum_{1 \le j \le n} j - 1 &= n(n-1)/2 \\
&< k \\
&= \sum_{1 \le i \le k} \mathrm{OR}_n(\alpha_i) \\
&\le \sum_i \sum_j \alpha_i(x_j) \\
&= \sum_j \sum_i \alpha_i(x_j).
\end{aligned}
$$

Therefore there exists j such that $j - 1 < \sum_i \alpha_i(x_j)$. Let $T_i = x_j$ for the least j values of i such that $\alpha_i(x_j) = 1$, and let $T_i = 1$ otherwise. Then $(T_1, \ldots, T_k)$ is an element of our proposed cover and $(T_1, \ldots, T_k)(\alpha) = \mathrm{OR}_n^{[k]}(\alpha)$.

Example 13. OR_k is pure $k+1$ for k in depth $k-1$ via the following cover:

$\mathrm{OR}_k(x_1,\ldots,x_{k-1})$	$\ldots$	$\mathrm{OR}_k(x_1,\ldots,x_{k-1})$
$\mathrm{OR}_k(x_1,\ldots,x_{k-2},x_k)$	$\ldots$	$\mathrm{OR}_k(x_1,\ldots,x_{k-2},x_k)$
	$\vdots$	
$\mathrm{OR}_k(x_2,\ldots,x_k)$	$\ldots$	$\mathrm{OR}_k(x_2,\ldots,x_k)$
1	$\ldots$	1

Consider k particular inputs to OR_k. Each nontrivial tree in a column gives the right answer unless the single bit it fails to examine is the only 1 in its input, in which case the last tree gives the right answer. Thus there is at most one wrong answer in every column of the table. Because there are more rows than columns, one row must be entirely correct.

3. Main Result

In this section we show that there is no nontrivial frequency computation in the decision tree model. As a corollary we show that $f^{[k]}$ cannot have a nontrivial cover of size k or less. Finally, this implies that at least $h+1$ help bits are necessary in order to assist with the computation of $f^{[2^h]}$.

Lemma 14. *Let $f_0,\ldots,f_{k-1}$ be boolean functions over $X_0,\ldots,X_{k-1}$, respectively. If there are k decision trees $T_0,\ldots,T_{k-1}$ over $\bigcup_{0\le i<k} X_i$ having depths $d_0,\ldots,d_{k-1}$, respectively, such that*

$$(\forall \alpha_0,\ldots,\alpha_{k-1})(\exists i)[T_i(\alpha_0,\ldots,\alpha_{k-1}) = f_i(\alpha_i)].$$

then

$$(\exists i)[f_i \in \mathrm{DEPTH}\,(d_i)].$$

Proof: We prove the lemma by strong induction on $k+\sum_{0\le i<k} d_i$. Let $f_0,\ldots,f_{k-1}$, $T_0,\ldots,T_{k-1}$, and $d_0,\ldots,d_{k-1}$ be as in the statement of the lemma. Assume that the statement of the lemma is true for all smaller values of $k+\sum_{0\le i<k} d_i$. We will prove that there exists i such that $f_i \in \mathrm{DEPTH}\,(d_i)$.

Case 1: $(\exists i)[d_i = 0]$. Wolog let $d_{k-1}=0$. If f_{k-1} is a constant function then $f_{k-1} \in \mathrm{DEPTH}\,(0)$ and we are done. Otherwise, there exists α_{k-1} such that $f_{k-1}(\alpha_{k-1}) \neq T_{k-1}(\alpha_0,\ldots,\alpha_{k-1})$. For $0 \le i < k-1$, obtain T_i' from T_i by applying the assignment α_{k-1}, i.e., if $\alpha_{k-1}(x)$ is defined then replace each node labeled x by its left subtree if $\alpha_{k-1}(x)=0$ or by its right subtree if $\alpha_{k-1}(x)=1$. Apply the inductive hypothesis to $T_0',\ldots,T_{k-2}'$ and $f_0,\ldots,f_{k-2}$, so $f_i \in \mathrm{DEPTH}\,(d_i)$ for some $i<k-1$.

Case 2: $(\forall i)[d_i > 0]$. If one of the trees T_i has an internal root, then renumber trees and functions so it is T_0. Otherwise every T_i has an external root. By a greedy algorithm find a sequence $i_0, \ldots, i_{m-1}$ such that $m \geq 1$, $0 = i_0 < \cdots < i_{m-1} < k$, and for $0 \leq j < m$ the root of T_{i_j} is an internal variable of $T_{(i_{j+1}) \bmod m}$. Renumber functions so that for $0 \leq j < m$ the root of T_j is an internal variable of $T_{(j+1) \bmod m}$.

Wolog let $X_i = \{x_{i1}, \ldots, x_{in_i}\}$ and $f_i = f_i(x_{i1}, \ldots, x_{in_i})$ for $0 \leq i < k$. Renumber variables so that for $0 \leq j < m$ the root of T_j is $x_{(j+1) \bmod m, 1}$. For $0 \leq j < m$, and $b \in \{0, 1\}$ define

$$f_j^b(x_{j2}, \ldots, x_{jn_j}) = f_j(b, x_{j2}, \ldots, x_{jn_j}).$$

For $0 \leq j < m$ and $\vec{b} = b(0) \cdots b(m-1) \in \{0,1\}^m$, define $T_j^{\vec{b}}$ to be the tree obtained from T_j by assigning $x_{01} \cdots x_{m-1,1} = \vec{b}$, i.e., by replacing each occurrence of the variable x_{i1} in T_j by its left subtree if $b(i) = 0$ or by its right subtree if $b(i) = 1$. Because we make an assignment to the root, the depth of $T_j^{\vec{b}}$ is at most $d_j - 1$.

For each $\vec{b} = b(0) \cdots b(m-1) \in \{0,1\}^m$ we apply the inductive hypothesis to the trees $T_0^{\vec{b}}, \ldots, T_{m-1}^{\vec{b}}, T_m, \ldots, T_{k-1}$ and the functions $f_0^{b(0)}, \ldots, f_{m-1}^{b(m-1)}, f_m, \ldots, f_{k-1}$. If one of these choices of $\vec{b}$ yields the conclusion that $f_j \in \mathrm{DEPTH}\,(d_j)$ for some $j \geq m$ then we are done.

Otherwise

$$(\forall b(0) \cdots b(m-1))(\exists j < m)\left[f_j^{b(j)} \in \mathrm{DEPTH}\,(d_j - 1)\right].$$

Since $b()$ is a Skolem function, that statement is equivalent to

$$(\exists j < m)(\forall b(j))\left[f_j^{b(j)} \in \mathrm{DEPTH}\,(d_j - 1)\right].$$

Therefore, $f_j \in \mathrm{DEPTH}\,(d)$. △

Theorem 15. $\mathrm{Freq}_d\,(1,k) = \mathrm{DEPTH}\,(d)$.

Proof: Obviously $\mathrm{DEPTH}\,(d) \subseteq \mathrm{Freq}_d\,(1,k)$. For the other containment, let $f \in \mathrm{Freq}_d\,(1,k)$ via $T_1, \ldots, T_k$. For $0 \leq i < k$, let $d_i = d$ and $f_i = f$. By Lemma 14, $f \in \mathrm{DEPTH}\,(d)$. △

Corollary 16. $\mathrm{Cover}_d\,(k,k) = \mathrm{DEPTH}\,(d)$.

Proof: Let f be a boolean function that is k for k in depth d via a cover as given below:

$$\begin{array}{c} \boxed{T_1^1 \;\cdots\; T_k^1} \\ \vdots \\ \boxed{T_1^k \;\cdots\; T_k^k} \end{array}$$

By the definition of cover,

$$(\forall \alpha_1, \ldots, \alpha_k)(\exists i)(\forall j)[T_j^i(\alpha_1, \ldots, \alpha_k) = f(\alpha_j)].$$

In particular,

$$(\forall \alpha_1, \ldots, \alpha_k)(\exists i)[T_i^i(\alpha_1, \ldots, \alpha_k) = f(\alpha_j)].$$

Therefore $f \in \mathrm{Freq}_d(1,k)$ via $T_1^1, \ldots, T_k^k$. By Theorem 15, $f \in \mathrm{DEPTH}(d)$. $\triangle$

Corollary 17. *If $f^{[k]}$ is computable by a depth-d forest with at most $\log_2 k$ help bits, then $f \in \mathrm{DEPTH}(d)$.*

Thus $\log_2 k$ help bits do not help you to compute $f^{[k]}$, and 1 help bit does not help you to compute $f^{[2]}$. (Of course with 1 help bit you can compute $f^{[1]}$ in depth 0, but that's not much of a feat.)

Proposition 18. $(\forall d \geq k-1)[\mathrm{DEPTH}(d) \subset \mathrm{PCover}_d(k+1,k)]$.

Proof: By Example 13, OR_k is pure $k+1$ for k in depth $k-1$. Let $f = \mathrm{OR}_k(x_1, \ldots, x_{k-1}, x_k \wedge \cdots \wedge x_{d+1})$. Then f is pure $k+1$ for k in depth d. By a simple adversary argument, $f \notin \mathrm{DEPTH}(d)$. $\triangle$

Proposition 19. $(\forall d)\left[\mathrm{PCover}_d\left(2^k-1,k\right) - \mathrm{DEPTH}(2d-1) \neq \emptyset\right]$.

Proof: By Example 12, OR_2 is pure 2^k-1 for k in depth 1. Therefore $(x_1 \wedge \cdots \wedge x_d) \vee (x_{d+1} \wedge \cdots \wedge x_{2d})$ is pure 2^k-1 for k in depth d. But $(x_1 \wedge \cdots \wedge x_d) \vee (x_{d+1} \wedge \cdots \wedge x_{2d})$ is not in $\mathrm{DEPTH}(2d-1)$ by a simple adversary argument. $\triangle$

Corollary 20. $(\forall d)\left[\mathrm{DEPTH}(d) \subset \mathrm{PCover}_d\left(2^k-1,k\right)\right]$. $\triangle$

Fix d. The proof of Proposition 19 gives us a nontrivial function f that is 2^k-1 for k for every k. In the Turing-machine and circuit models of bounded queries, we have many examples of nontrivial functions f that are $k+1$ for k for every k [5, 2, 3, 4]. In contrast, the proof of Proposition 18 gives us infinitely many different functions f_k such that f_k is $k+1$ for k. Using a very nice result due to Nisan et al. [18, Lemma 4.1], we will show that no single nontrivial function f can be $k+1$ for k for every k.

Lemma 21 (NRS Lemma). *Let f be a Boolean function, d a nonnegative integer, and $p(f,d)$ the maximum real number p such that there is a distribution on depth-d decision trees that approximates f with probability p. Then*

$$(\forall k)\left[f \in \mathrm{PCover}_d\left(\frac{nk\ln 2}{p(f,d)^k}, k\right) - \mathrm{Cover}_d\left(\frac{1}{p(f,d)^k} - 1, k\right)\right]. \quad \triangle$$

Proposition 22. *Let $c(k) = 2^{o(k)}$. If $f \in \mathrm{Cover}_d(c(k),k)$ for infinitely many k, then $f \in \mathrm{DEPTH}(d)$.*

Proof: By the NRS Lemma, we have for infinitely many k

$$f \in \mathrm{Cover}_d(c(k),k) - \mathrm{Cover}_d\left(\frac{1}{p(f,d)^k} - 1, k\right)$$

so $\frac{1}{p(f,d)^k} \leq c(k) = 2^{o(k)}$. Therefore $p(f,d) = 1$, so $f \in \mathrm{DEPTH}(d)$. $\triangle$

4. On Pure Families

Lemma 23. *For* $1 \leq a \leq m$,

$$\mathrm{PCover}_d\,(m,k) \subseteq \mathrm{Prob}_d\,(a,m) \cup \mathrm{PCover}_d\,(a-1,k-1)\,.$$

Proof: Let f be pure m for k in depth d. Consider a cover for $f^{[k]}$ arranged in a grid as usual. If a answers in the first column are always correct then $f \in \mathrm{Prob}_d\,(a,m)$ via the trees in the first column. Otherwise, fix an input x on which the first column errs at least $m-a+1$ times. Delete the corresponding $m-a+1$ rows and the first column, to obtain a cover for $f^{[k-1]}$. △

Lemma 24. *Let* $k, a \geq 1$.

$$\mathrm{Prob}_d\,(k,k+a) \subseteq \mathrm{PCover}_d\,(k+a, \lceil k/a \rceil)\,.$$

Proof: Let $\ell = \lceil k/a \rceil$. Let f be k out of $k+a$ via trees $T_1, \ldots, T_{k+a}$. For each input α to f, at most a of the trees T_i satisfy $T_i(\alpha) \neq f(\alpha)$. Since $k+a > a\ell$, for each input $\alpha_1, \ldots, \alpha_\ell$ to $f^{[\ell]}$,

$$(\exists i \leq k+a)(\forall j \leq \ell)[T_i(\alpha_j) = f(\alpha_j)].$$

Thus, the following is a size-$(k+a)$, depth-d pure cover for $f^{[\ell]}$:

$$\begin{array}{c} \boxed{T_1 \quad \ldots \quad T_1} \\ \vdots \\ \boxed{T_{k+a} \quad \ldots \quad T_{k+a}} \end{array}$$

Theorem 25. $\mathrm{Prob}_d\,(k,k+1) = \mathrm{PCover}_d\,(k+1,k)$.

Proof: By Lemma 23 with $a = k$,

$$\begin{aligned} \mathrm{PCover}_d\,(k+1,k) &\subseteq \mathrm{Prob}_d\,(k,k+1) \cup \mathrm{PCover}_d\,(k-1,k-1) \\ &= \mathrm{Prob}_d\,(k,k+1) \cup \mathrm{DEPTH}\,(d) \qquad \text{by Corollary 16} \\ &= \mathrm{Prob}_d\,(k,k+1)\,. \end{aligned}$$

The converse follows from Lemma 24 with $a = 1$. △

Corollary 26. $\mathrm{Prob}_d\,(2,3) = \mathrm{PCover}_d\,(3,2)$.

Proof: Let $k = 2$ in Theorem 25. △

Corollary 27. $\mathrm{PCover}_d\left(2^k - 1, k\right) \subseteq \bigcup_{j \leq k} \mathrm{Prob}_d\left(2^{j-1}, 2^j - 1\right)$.

Proof: This follows from Lemma 23 by an easy induction. △

Definition 28 (Depth-d Reductions and Completeness).

- $f(x_1, \ldots, x_a)$ is depth-d reducible to $g(x_1, \ldots, x_b)$ if there exist depth-d decision trees $T_1, \ldots, T_b$ over $\{x_1, \ldots, x_a\}$ such that

$$f(x_1, \ldots, x_a) = g(T_1(x_1, \ldots, x_a), \ldots, T_b(x_1, \ldots, x_a)).$$

- f is *complete* for $\mathrm{Cover}\,(m,k)$ if $f \in \mathrm{Cover}_1\,(m,k)$ and, for every d, every function in $\mathrm{Cover}_d\,(m,k)$ is depth-d reducible to f.

- f is *complete* for $\mathrm{PCover}\,(m,k)$ if $f \in \mathrm{PCover}_1\,(m,k)$ and, for every d, every function in $\mathrm{PCover}_d\,(m,k)$ is depth-d reducible to f.

Corollary 26 implies the existence of a complete problem for $\mathrm{PCover}\,(3,2)$.

Corollary 29. MAJ_3 *is complete for* $\mathrm{PCover}\,(3,2)$. △

Thus $\mathrm{PCover}_1\,(3,2) \subseteq \mathrm{PCover}_1\,(m,k)$ if and only if $\mathrm{MAJ}_3 \in \mathrm{PCover}_1\,(m,k)$. Theorem 25 implies the following normal form for size-$(k+1)$ pure covers for $f^{[k]}$:

$$\begin{array}{c} \boxed{T_1 \quad \ldots \quad T_1} \\ \vdots \\ \boxed{T_{k+1} \quad \ldots \quad T_{k+1}} \end{array}$$

Example 30. Since this size-3 pure family covers $\mathrm{OR}_2^{[2]}(x,y)$,

x	y
y	1
1	x

the family below also covers $\mathrm{OR}_2^{[2]}(x,y)$:

x	x
y	y
1	1

Lemma 31. $\mathrm{Prob}_d\,(k,m) \subseteq \mathrm{Prob}_d\,(k-1, m-1)$.

Proof: Given m trees that witness the fact that f is k out of m, any $m-1$ trees among them witness the fact that f is $k-1$ out of $m-1$. △

(The converse to the lemma above is easily seen to be false.)

Corollary 32. $\mathrm{PCover}_d\,(3,2) - \mathrm{PCover}_{3d-1}\,(7,4) \neq \emptyset$.

Proof:

$$\begin{aligned}
\mathrm{PCover}_{3d-1}(7,4) &\subseteq \mathrm{Prob}_{3d-1}(5,7) \cup \mathrm{PCover}_{3d-1}(4,3) \qquad \text{by Lemma 23 with } a=5 \\
&= \mathrm{Prob}_{3d-1}(5,7) \cup \mathrm{Prob}_{3d-1}(3,4) \qquad \text{by Theorem 25} \\
&= \mathrm{Prob}_{3d-1}(5,7) \cup (\mathrm{Prob}_{3d-1}(3,4) \cap \mathrm{Prob}_{3d-1}(2,3)) \qquad \text{by Lemma 31} \\
&\subseteq \mathrm{Prob}_{3d-1}(5,7) \cup \mathrm{Prob}_{3d-1}(5,7) \\
&= \mathrm{Prob}_{3d-1}(5,7)
\end{aligned}$$

That means every function in $\mathrm{PCover}_{3d-1}(7,4)$ is recognized with probability at least $\frac{5}{7}$ by a probabilistic depth-$(3d-1)$ decision tree algorithm. But there is a function in $\mathrm{PCover}_d(3,2)$ that does not have such an algorithm, namely, $f = \mathrm{MAJ3}(x_1 \oplus \cdots \oplus x_d, x_{d+1} \oplus \cdots \oplus x_{2d}, x_{2d+1} \oplus \cdots \oplus x_{3d})$. Why? Consider the set of all inputs where the values $x_1 \oplus \cdots \oplus x_d$, $x_{d+1} \oplus \cdots \oplus x_{2d}$, and $x_{2d+1} \oplus \cdots \oplus x_{3d}$ are not mutually equal. Any depth-$(3d-1)$ decision tree answers at most two-thirds of those problem instances correctly, so no probabilistic depth-$(3d-1)$ algorithm can be correct with probability greater than two-thirds. $\triangle$

Theorem 33. $(\forall k \geq 2)[\mathrm{PCover}_d(2k+1, k+1) \subseteq \mathrm{PCover}_d(2k-1, k)]$.

Proof:

$$\begin{aligned}
&\mathrm{PCover}_d(2k+1, k+1) \\
&\quad\subseteq \mathrm{Prob}_d(2k, 2k+1) \cup \mathrm{PCover}_d(2k-1, k) \qquad \text{by Lemma 23 with } a = 2k \\
&\quad\subseteq \mathrm{Prob}_d(2k-2, 2k-1) \cup \mathrm{PCover}_d(2k-1, k) \qquad \text{by Lemma 31} \\
&\quad= \mathrm{PCover}_d(2k-1, 2k-2) \cup \mathrm{PCover}_d(2k-1, k) \qquad \text{by Theorem 25} \\
&\quad= \mathrm{PCover}_d(2k-1, k).
\end{aligned}$$

$\triangle$

Thus $\mathrm{PCover}_d(3,2) \supseteq \mathrm{PCover}_d(5,3) \supseteq \mathrm{PCover}_d(7,4) \supseteq \cdots$. By Corollary 32, for each d one of the first two containments must be proper. In the next section, we will see that in fact $\mathrm{PCover}_d(3,2) = \mathrm{PCover}_d(5,3) \supset \mathrm{PCover}_d(7,4)$ for every d. What's more, for infinitely many k, there exists $d \leq m$ such that $\mathrm{PCover}_d(2k-1, k-1) \supset \mathrm{PCover}_d(2k+1, k)$. Why? Assume for the sake of contradiction that $\mathrm{PCover}_d(2d-1, d) = \mathrm{PCover}_d(2d+1, d+1) = \mathrm{PCover}_d(2d+3, d+2) = \cdots$. By Proposition 18 there exists $f \in \mathrm{PCover}_d(2d-1, d) - \mathrm{DEPTH}(d)$. Then, by assumption, $f \in \mathrm{PCover}_d(2k-1, k)$ for all $k > d$. But, by Proposition 22, $f \in \mathrm{DEPTH}(d)$, a contradiction.

Corollary 34. $(\forall k \geq 0)[\mathrm{PCover}_d(2k+1, k+1) \subseteq \mathrm{PCover}_d(3,2)]$.

Proof: If $k = 0$ then the containment is obvious. If $k \geq 1$, then the containment follows from Theorem 33 by transitivity. $\triangle$

Corollary 34 implies a limit on size–depth tradeoffs for pure covers:

Corollary 35. $\mathrm{PCover}_d\,(2k-1,k) \subseteq \mathrm{DEPTH}\,(3d)$.

Proof: Let $f \in \mathrm{PCover}_d\,(2k-1,k)$.

$$\begin{array}{rcll} \mathrm{PCover}_d\,(2k-1,k) & \subseteq & \mathrm{PCover}_d\,(3,2) & \text{by Corollary 34} \\ & = & \mathrm{Prob}_d\,(2,3) & \text{By Corollary 26.} \end{array}$$

Therefore, f is the majority of three depth-d decision trees, so $f \in \mathrm{DEPTH}\,(3d)$. $\triangle$

It is not known whether there is a similar limit on size–depth tradeoffs for general covers.

5. On Constructing a Cover

Nisan et al. [18] proved nearly tight existential bounds on the size of a cover for $f^{[k]}$. In this section we show how to construct a cover for $\mathrm{MAJ}^{[k]}_{2m+1}$ that is less than $\log_2 e$ times as large as Nisan et al's upper bound (so our constructive bound is almost as good as the existential bound) and less than $2km$ times as large as Nisan et al's lower bound (so our cover is close to optimal in size). Then we transform our cover for $\mathrm{MAJ}^{[k]}_{2m+1}$ into covers for other functions.

Let g be any function with variable set X. The problem of finding a minimum cover for g, using only forests in a given class C, is equivalent to the problem of finding a minimum vertex cover for the hypergraph $H(C,g)$ constructed as follows:

- The vertex set is the set C of forests.
- For each assignment α to X, there is one hyperedge consisting of all vertices F such that $F(\alpha) = g(\alpha)$.

Now let $g = \mathrm{MAJ}^{[k]}_{2m+1}(x_1,\ldots,x_{2m+1})$, and let C be the class of pure depth-1 forests in which every tree is x_i for some i between 1 and $2m+1$. Wolog, we can restrict our attention to assignments that assign exactly m 0's and $m+1$ 1's to each input $(x_1,\ldots,x_{2m+1})$ (by our choice of C, a cover for those assignments is a cover for all assignments). In particular, $g(\alpha) = 1$ for every assignment α we will consider.

The hypergraph $H = H(C,g)$ consists of $(2m+1)^k$ vertices and $\binom{2m+1}{m}^k$ hyperedges. The degree, d, of the hypergraph is $\binom{2m}{m}^k$. Each hyperedge consists of exactly $(m+1)^k$ vertices. We give each vertex v the weight $t_v = \frac{1}{(m+1)^k}$. Then for every hyperedge e,

$$\Sigma_{v\in e} t_v = 1.$$

The weighting function t_v is a fractional cover of H of size $\sum_v t_v = \left(\frac{2m+1}{m+1}\right)^k$. By Lovasz's theorem on fractional covers [16], H has a vertex cover whose size is less than $(1 + \log_2 d)$ times the size of the fractional cover, i.e., H has a vertex cover of size less than

$$\left(1 + k \log_2 \binom{2m}{m}\right) \left(\frac{2m+1}{m+1}\right)^k \; < \; 2km \left(2 - \frac{1}{m+1}\right)^k .$$

For all sufficiently large k, that bound is less than 2^k. What's more, Lovasz [16] gives a greedy algorithm that constructs the desired cover.

Corollary 36. $\quad \mathrm{MAJ}_{2m+1} \in \mathrm{PCover}_1 \left(2km \left(2 - \frac{1}{m+1}\right)^k, k\right).$
Furthermore, the desired cover can be constructed by a greedy algorithm.

In contrast, the NRS Lemma provides a slightly better bound for the majority function, $k(2m+1)\ln 2 \left(2 - \frac{1}{m+1}\right)^k$, but because their lemma is proved by the probabilistic method it does not tell you how to construct the cover.

Nisan et al. [18] discovered an important connection between covers and the sign-degree of a function. We exploit this connection now in order to pure covers.

For the remainder of this section we will let -1 represent false and 1 represent true.

Definition 37. Let f be an n-variable boolean function.

- We say that an n-variable real polynomial p *sign-represents* f if

  $$(\forall x)[\mathrm{sgn}(p(x)) = f(x)],$$

  where $\mathrm{sgn}(z) = 1$ if $z > 0$, -1 otherwise.

- The *sign-degree* of f (denoted $\deg^s(f)$) is the minimum degree of a polynomial that sign-represents f.

The next proposition is due to [18]. Here we give a constructive proof, which provides a way of constructing covers.

Proposition 38. $\quad (\exists m < 2^k)[f \in \mathrm{PCover}_d\,(m, k)] \iff d \geq \deg^s(f).$

Proof: Let p be a minimum-degree polynomial such that $f(x_1, \ldots, x_n) = \mathrm{sgn}(p(x_1, \ldots, x_n))$. Then

$$p(x_1, \ldots, x_n) = q_1 t_1 + \cdots + q_\ell t_\ell$$

where $q_1, \ldots, q_\ell$ are rational numbers and $t_1, \ldots, t_\ell$ are products of at most $\deg^s(f)$ variables. Wolog, let $q_1, \ldots, q_\ell$ be integers (clearing denominators preserves the sign). Then

$$\mathrm{sgn}(p(x_1, \ldots, x_n)) = \mathrm{MAJ}_u(t'_1, \ldots, t'_u) \tag{1}$$

where each term t_i appears q_i times as an argument to MAJ_u. Wolog, u is odd (otherwise insert a constant parameter). Let $2v+1=u$.

Thus f is depth-$\deg^s(f)$ reducible to MAJ_{2v+1}. Therefore,

$$f \in \mathrm{PCover}_{\deg^s(f)}\left(2kv\left(2-\frac{1}{v+1}\right)^k, k\right)$$

and the promised cover can be constructed by a greedy algorithm.

For the other direction, assume that f is pure m for k in depth d where $m < 2^k$. By Corollary 27, there exists m' such that f is $\lfloor m'/2 \rfloor + 1$ out of m'. Thus there are depth-d trees $T_1, \ldots, T_{m'}$ such that

$$f(x) = \mathrm{MAJ}_{m'}(T_1(x), \ldots, T_{m'}(x)).$$

Therefore

$$f(x) = \mathrm{sgn}(T_1(x) + \cdots + T_{m'}(x)).$$

Since each T_i has depth at most d, T_i is exactly equal to a polynomial of depth at most d. Therefore $\deg^s(f) \leq d$. $\triangle$

6. Equivalences

In this section we will show that depth-1 covers for MAJ_3 and for OR_2 have exactly the same size. One consequence is that $\mathrm{PCover}_d(3,2) = \mathrm{PCover}_d(5,3)$.

Definition 39. Let f and g be boolean functions. We say that f is *equivalent* to g (denoted $f \equiv g$) if

$$(\forall m)(\forall k)[f \in \mathrm{PCover}_1(m,k) \iff g \in \mathrm{PCover}_1(m,k)].$$

Lemma 40. *Let $f(x_1, \ldots, x_n)$ be any boolean function and*

$$G_f(x_1, \ldots, x_n, z) = (z \wedge f(x_1, \ldots, x_n)) \ \vee \ (\neg z \wedge \neg f(\neg x_1, \ldots, \neg x_n)).$$

Then $G_f \equiv f$.

Proof: Because $f(x_1, \ldots, x_n) = G_f(x_1, \ldots, x_n, 1)$,

$$G_f \in \mathrm{PCover}_1(m,k) \Rightarrow f \in \mathrm{PCover}_1(m,k).$$

For the converse, let $\mathcal{F}$ be a depth-1, size-m pure cover for $f^{[k]}(x_{11}, \ldots, x_{1n}, z_1, \ldots, x_{k1}, \ldots, x_{kn}, z_k)$. For each $F \in \mathcal{F}$, construct F^* by replacing each tree in F that computes the constant function 1 by the depth-1 tree computing z, and each tree in F that computes the constant function 0 by the depth-1 tree computing $\neg z$. For each assignment α, define

$$\alpha'(x_{ij}) = \neg\alpha(x_{ij}) \oplus \alpha(z_i)$$

for all i, j. Because $\mathcal{F}$ is a pure cover for $f^{[k]}$, there exists a forest $F \in \mathcal{F}$ such that $F(\alpha') = f^{[k]}(\alpha')$. But then $F^*(\alpha) = G_f^{[k]}(\alpha)$, so $\{F^* : F \in \mathcal{F}\}$ is a depth-1, size-m pure cover for $G_f^{[k]}$. $\triangle$

In particular $\mathrm{OR}_n \equiv M_{n+1}$ where

$$M_{n+1}(x_1, \ldots, x_n, z) = \begin{cases} \mathrm{OR}_n(x_1, \ldots, x_n) & \text{if } z = 1 \\ \mathrm{AND}_n(x_1, \ldots, x_n) & \text{if } z = 0 \end{cases}$$

We note a nice special case:

Theorem 41. $\mathrm{OR}_2 \equiv \mathrm{MAJ}_3$.

Corollary 42. $(\forall d) \left[\mathrm{PCover}_d\left(2^k - 1, k\right) - \mathrm{DEPTH}\,(3d - 1) \neq \emptyset\right]$.

Proof: By Example 12, OR_2 is pure $2^k - 1$ for k in depth 1. Therefore, by Theorem 41, MAJ_3 is pure $2^k - 1$ for k in depth 1. Therefore $\mathrm{MAJ}_3(x_1 \wedge \cdots \wedge x_d, x_{d+1} \wedge \cdots \wedge x_{2d}, x_{2d+1} \wedge \cdots \wedge x_{3d})$ is pure $2^k - 1$ for k in depth d. But $\mathrm{MAJ}_3(x_1 \wedge \cdots \wedge x_d, x_{d+1} \wedge \cdots \wedge x_{2d}, x_{2d+1} \wedge \cdots \wedge x_{3d})$ is not in $\mathrm{DEPTH}\,(3d - 1)$ by a simple adversary argument. $\triangle$

Corollary 43. $\mathrm{MAJ}_3 \in \mathrm{PCover}_1\,(5, 3)$.

Proof: By Example 11, $\mathrm{OR}_2^{[3]}$ in $\mathrm{PCover}_1\,(5, 3)$. By Theorem 41, $\mathrm{OR}_2 \equiv \mathrm{MAJ}_3$. $\triangle$

Example 44. Here is the depth-1, size-5 pure cover for $\mathrm{MAJ}_3^{[3]}$ implied by the result above:

x	x	x
z	y	y
y	z	y
y	y	z
z	z	z

Corollary 45. $\mathrm{PCover}_d\,(3, 2) = \mathrm{PCover}_d\,(5, 3)$.

Proof: If f is pure 3 for 2 then f is 2 out of 3. Let T_1, T_2, T_3 be the appropriate pure trees. Then, $f(x) = \mathrm{MAJ}_3(T_1(x), T_2(x), T_3(x))$. By Corollary 43, $\mathrm{MAJ}_3^{[3]}(x, y, z)$ has a depth-1, size-5 pure cover $\mathcal{F}$. A depth-d, size-5 pure cover for $f^{[3]}$ is obtained from $\mathcal{F}$ by replacing x by T_1, y by T_2 and z by T_3. Thus $\mathrm{PCover}_d\,(3, 2) \subseteq \mathrm{PCover}_d\,(5, 3)$.

The converse follows from Theorem 33. $\triangle$

Now we know that

$$\mathrm{PCover}_d\,(3, 2) = \mathrm{PCover}_d\,(5, 3) \supset \mathrm{PCover}_d\,(7, 4) \supseteq \mathrm{PCover}_d\,(9, 5) \supseteq \mathrm{PCover}_d\,(11, 6) \supseteq \cdots.$$

For infinitely many k, there exists d such that $\mathrm{PCover}_d\,(2k - 1, k) \supset \mathrm{PCover}_d\,(2k + 1, k + 1)$. However, it is unknown whether $\mathrm{PCover}_1\,(7, 4) = \mathrm{PCover}_1\,(9, 5) = \mathrm{PCover}_1\,(11, 6) = \cdots = \mathrm{DEPTH}\,(1)$.

7. Open questions

When is it true that

- $\mathrm{DEPTH}(d) \subset \mathrm{PCover}_d(m,k)$?
- $\mathrm{DEPTH}(d) \subset \mathrm{Cover}_d(m,k)$?
- $\mathrm{Cover}_d(m,k) = \mathrm{PCover}_d(m,k)$?
- $\mathrm{Cover}_d(m,k) \subseteq \mathrm{Cover}_d(m',k')$?
- $\mathrm{PCover}_d(m,k) \subseteq \mathrm{PCover}_d(m',k')$?
- $\mathrm{Prob}_d(m,k) \subseteq \mathrm{Prob}_d(m',k')$?
- $\mathrm{Cover}(m,k)$ has complete problems?

Acknowledgments

The authors are very grateful to Moshe Lewenstein for his help in preparing this paper and to Bill Gasarch for his advice on the exposition.

References

[1] A. Amir, R. Beigel, and W. I. Gasarch. Some connections between bounded query classes and nonuniform complexity. In *Proceedings of the 5th Annual Conference on Structure in Complexity Theory*, pages 232–243, 1990.

[2] A. Amir and W. I. Gasarch. Polynomial terse sets. *Inf. & Comp.*, 77:37–56, Apr. 1988.

[3] R. Beigel. A structural theorem that depends quantitatively on the complexity of SAT. In *Proceedings of the 2nd Annual Conference on Structure in Complexity Theory*, pages 28–32. IEEE Computer Society Press, June 1987.

[4] R. Beigel, W. I. Gasarch, J. T. Gill, and J. C. Owings. Terse, superterse, and verbose sets. *Inf. & Comp.*, 103:68–85, 1993.

[5] R. Beigel, W. I. Gasarch, and L. Hay. Bounded query classes and the difference hierarchy. *Archive for Mathematical Logic*, 29(2):69–84, Dec. 1989.

[6] R. Beigel, W. I. Gasarch, and E. Kinber. Frequency computation and bounded queries. In *Proceedings of the 10th Annual Conference on Structure in Complexity Theory*, pages 125–132, 1995. Also appeared in Structures '95.

[7] R. Beigel, M. Kummer, and F. Stephan. Approximable sets. *Inf. & Comp.*, 120(2):304–314, 1995.

[8] R. Beigel, M. Kummer, and F. Stephan. Quantifying the amount of verboseness. *Inf. & Comp.*, 118(1):73–90, 1995. A preliminary version appeared in: *Lecture Notes for Computer Science ("Logic at Tver")*, Vol. 620, pp. 21–32, 1992.).

[9] J. Cai. Lower bounds for constant depth circuits in the presence of help bits. *IPL*, 36:79–84, 1990. Also appeared in FOCS 1989.

[10] J. Cai and L. A. Hemachandra. Enumerative counting is hard. *Inf. & Comp.*, 82(1):34–44, July 1989.

[11] A. N. Degtev. On (m, n)-computable sets. In D. Moldavanskij, editor, *Algebraic Systems*, pages 88–99. Ivanova Gos. Univ., 1981. Russian (MR 86b:03049).

[12] V. Harizanov, M. Kummer, and J. C. Owings. Frequency computations and the cardinality theorem. *JSL*, 57(2):682–687, June 1992.

[13] E. B. Kinber. On frequency calculations of general recursive predicates. *Soviet Math. Doklady*, 13:873–876, 1972.

[14] M. Kummer. A proof of Beigel's cardinality conjecture. *JSL*, 57(2):677–681, June 1992.

[15] M. Kummer and F. Stephan. Recursion theoretic properties of frequency computation and bounded queries. *Inf. & Comp.*, 120:59–77, 1995.

[16] L. Lóvasz. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[17] R. McNaughton. The theory of automata, a survey. *Advances of Computers*, 2:379–421, 1961.

[18] N. Nisan, S. Rudich, and M. Saks. Products and help bits in decision trees. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, pages 318–329, 1994. To appear in SICOMP.

[19] G. F. Rose. An extended notion of computability. *Abstr. Intern. Congr. for Logic, Meth., and Phil. of Science*, 1960.

[20] B. A. Trakhtenbrot. On the frequency computation of functions. *Algebra i Logika*, 2, 1964. Russian.