k^+ Decision Trees*
(Extended Abstract)

James Aspnes[1], Eric Blais[2], Murat Demirbas[3], Ryan O'Donnell[2], Atri Rudra[3], and Steve Uurtamo[3]

[1] Department of Computer Science, Yale University
New Haven, CT 06520.
aspnes@cs.yale.edu
[2] Department of Computer Science, Carnegie Mellon University
Pittsburgh, PA 15213.
{eblais,odonnell}@cs.cmu.edu
[3] Department of Computer Science and Engineering, University at Buffalo
State University of New York, Buffalo, NY, 14260.
{demirbas,atri,uurtamo}@buffalo.edu

Abstract. Consider a wireless sensor network in which each sensor has a bit of information. Suppose all sensors with the bit 1 broadcast this fact to a basestation. If zero or one sensors broadcast, the basestation can detect this fact. If two or more sensors broadcast, the basestation can only detect that there is a "collision." Although collisions may seem to be a nuisance, they can in some cases help the basestation compute an aggregate function of the sensors' data.

Motivated by this scenario, we study a new model of computation for boolean functions: the 2^+ *decision tree*. This model is an augmentation of the standard decision tree model: now each internal node queries an arbitrary *set* of literals and branches on whether 0, 1, or at least 2 of the literals are true. This model was suggested in a work of Ben-Asher and Newman but does not seem to have been studied previously.

Our main result shows that 2^+ decision trees can "count" rather effectively. Specifically, we show that zero-error 2^+ decision trees can compute the threshold-of-t symmetric function with $O(t)$ expected queries (and that $\Omega(t)$ is a lower bound even for two-sided error 2^+ decision trees). Interestingly, this feature is not shared by 1^+ decision trees. Our result implies that the natural generalization to k^+ decision trees does not give much more power than 2^+ decision trees. We also prove a lower bound of $\tilde{\Omega}(t) \cdot \log(n/t)$ for the *deterministic* 2^+ complexity of the threshold-of-t function, demonstrating that the randomized 2^+ complexity can in some cases be unboundedly better than deterministic 2^+ complexity.

Finally, we generalize the above results to arbitrary symmetric functions, and we discuss the relationship between k^+ decision trees and other complexity notions such as decision tree rank and communication complexity.

* James Aspnes is supported in part by NSF grant CCF-0916389. Ryan O'Donnell is supported in part by NSF grants CCF-0747250 and CCF-0915893, a Sloan fellowship, and an Okawa fellowship. Atri Rudra and Steve Uurtamo are supported in part by NSF CAREER grant CCF-0844796.

1 Introduction

The motivation for our work comes from monitoring applications in wireless sensor networks. Consider the scenario where n sensors communicate directly with a basestation (i.e., a "single-hop" network). Each sensor contains one bit of information (e.g., "Is the temperature more than 70°F?") and the basestation wants to compute some aggregate function over this information (e.g., "Is the temperature more than 70°F for at least 10 of the sensors?"). How efficiently can the basestation compute the aggregate function? The naive way to compute the aggregate function is to query each sensor individually resulting in as many queries as the number of sensors n in the worst case.

We observe that a better solution is to use the broadcast primitive available in wireless networks. With a single broadcast message, the basestation may simultaneously ask a group of sensors if their bit is 1, and switches to the listening mode. There are three possible scenarios: either 0, 1, or at least 2 sensors reply. In the first two cases, the basestation can detect that exactly 0 or exactly 1 sensors in the group have replied. In the third case, there will be a collision in the sensors' replies. Conventional wisdom says that collisions are bad, and in fact protocols in wireless networks try to disambiguate collisions. In this scenario, however, collisions provide useful information: the basestation gets a quick feedback affirming that at least 2 sensors have replied in the queried group.

Such a quick in-network feedback collection primitive has many uses in wireless sensor network applications. For example for intrusion detection applications [2], a clusterhead can employ this primitive to quickly check if a threshold number of sensors have detected an intruder. The clusterhead may initiate the more costly actions to localize and classify the intruder, only if a threshold number of sensors detect the phenomena (i.e., after it is assured that this is not a false-positive due to wind, low-quality sensors, etc.).

The receiver side collision detection required for this primitive can easily be implemented in the existing sensor radios. Indeed, there has been recent work in using existing wireless sensor nodes to detect collisions and using this capability to design more efficient protocols [7, 8, 11].

Next, we present our theoretical framework for studying the problem above.
Decision trees. Decision trees provide an elegant framework for studying the complexity of boolean functions. The internal nodes of a decision tree are associated with *tests* on the input; the branches leaving a node correspond to the outcomes of the associated test; and, the leaves of the tree are labeled with output values. The main parameter of interest is the *depth* of the decision tree; i.e., the maximum number of tests made over all inputs. The decision tree complexity of a particular boolean function is defined to be the minimum of this parameter over all decision trees computing the function.

We can define different decision tree models by restricting the set of tests that can be performed at each internal node. In the *simple* decision tree model, each test queries the value of a single bit of the input. This standard model is extremely well-studied in theoretical computer science; see e.g. the survey of Buhrman and de Wolf [6]. Other models include *linear* decision trees, where

the tests are signs of linear forms on the bits (see, e.g., [9]); *algebraic* decision trees, the generalization to signs of low-degree polynomials (see, e.g., [4]); k-AND decision trees [5], where the tests are ANDs of up to k literals; k-bounded decision trees [15], the generalization to arbitrary functions of up to k bits; and $\mathbb{F}_2$-linear decision trees [13], where the tests are parities of sets of input bits.

Closer to the interests of the present article are models where the tests are *threshold functions* of the input bits. When the tests can be ORs of input variables, the model is known as *combinatorial group testing* (see the book [10]). When the tests can count and branch on the number of inputs in any subset of variables, the model is connected to that of *combinatorial search* (see the book [1]). Finally, when the tests allowed are ORs of any subset of input *literals*, we have the decision tree model studied by Ben-Asher and Newman [3]. We call this last model the *1^+ decision tree* model.

In this article, we initiate the study of the *2^+ decision tree* model. If the algorithm at the basestation does not try to avoid collisions but instead uses them to determine when at least 2 sensors have replied, the complexity of computing the aggregate function is determined by the 2^+ decision tree complexity of the function. The tests in this model are on arbitrary subsets of the n input literals, and the branches correspond to the cases of either 0, 1, or at least 2 literals in the subset being true. (We will give a formal definition in Section 2.1.) We also introduce and examine the *k^+ decision tree* model, a natural generalization of the 2^+ model in which the branches correspond to 0, 1, 2, ..., $k-1$, or at least k literals being true.

A similar motivation for 2^+ decision trees appears in the work of Ben-Asher and Newman [3]. They were concerned with the PRAM model of parallel computation with n processors and a one-bit CRCW memory cell. This led naturally to the 1^+ model of computation, which Ben-Asher and Newman studied. The authors also mentioned that an Ethernet channel scenario — say, a single bus Ethernet network where the controller can detect any collisions in the network — yields a computational model equivalent to our 2^+ decision trees, but left the study of this model as an open problem.

Organization of the article. We defer the statement of our main results to Section 3, in order to first introduce formal definitions of our models. We prove our main results in Sections 4 and 5, and we present other results from the full version of this article in Section 6. Due to lack of space, the missing details including all the omitted proofs will appear in the full version of the paper.

2 Preliminaries

2.1 Definitions

In this article, we are concerned with boolean functions; i.e., functions of the form $f : \{0,1\}^n \to \{0,1\}$. We write a typical input as $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, and write $|x|$ for its Hamming weight, namely $\sum_{i=1}^n x_i$. We also use the notation $[n] = \{1, \ldots, n\}$ and $\log(m) = \max\{\log_2(m), 1\}$.

A boolean function $f : \{0,1\}^n \to \{0,1\}$ is *monotone* if $f(x) \geq f(y)$ whenever $x \geq y$ coordinate-wise. A function f is *(totally) symmetric* if the value of $f(x)$ is determined by $|x|$. When f is symmetric, we write $f_0, f_1, \ldots, f_n \in \{0,1\}$ for the values of the function on inputs of Hamming weight $0, 1, \ldots n$, respectively. The functions which are both monotone and symmetric are the *threshold* functions. Given $0 \leq t \leq n+1$, the *t-threshold* function $T_n^t : \{0,1\}^n \to \{0,1\}$ is defined by $T_n^t(x) = 1$ iff $|x| \geq t$.

For an arbitrary symmetric function f we recall the integer parameter $\Gamma(f)$, first introduced by Paturi [17], and related to the longest interval centered around $n/2$ on which f is constant:

$$\Gamma(f) = \min_{0 \leq \ell \leq \lceil n/2 \rceil} \{\ell : f_\ell = f_{\ell+1} = \cdots = f_{n-\ell}\}.$$

E.g., for the threshold functions we have $\Gamma(T_n^t) = \min\{t, n+1-t\}$.

k^+ decision trees. Let $1 \leq k \leq n$ be integers. A k^+ *decision tree* T over n-bit inputs is a tree in which every leaf has a label from $\{0,1\}$, every internal node is labeled with two disjoint subsets $Q_{\text{pos}}, Q_{\text{neg}} \subseteq [n]$, and the internal nodes have $k+1$ outgoing edges labeled $0, 1, \ldots, k-1$, and k^+. Every internal node is also called a *query*; the corresponding sets Q_{pos} and Q_{neg} are called the *positive query set* and *negative query set*. Given a boolean input $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, the computation of T on x begins at the root of T. If that node is labeled by $(Q_{\text{pos}}, Q_{\text{neg}})$, computation proceeds along the edge labeled $0, 1, \ldots, k-1$, or k^+ according to Hamming weight of the literal set $\{x_i : i \in Q_{\text{pos}}\} \cup \{\overline{x}_j : j \in Q_{\text{neg}}\}$; i.e., $\sum_{i \in Q_{\text{pos}}} x_i + \sum_{j \in Q_{\text{neg}}} \overline{x}_j$. The label k^+ has the interpretation "at least k." The computation of T on x then proceeds recursively at the resulting child node. When a leaf node is reached, the tree's output on x, denoted by $T(x)$, is the label of the leaf node. T is said to *compute* (or *decide*) a function $f : \{0,1\}^n \to \{0,1\}$ if and only if $T(x) = f(x)$ for all $x \in \{0,1\}^n$. The *cost* of T on x, denoted $\text{cost}(T, x)$, is the length of the path traced by the computation of T on x. The *depth* of the tree T is the maximum cost over all inputs. The *deterministic k^+ decision tree complexity* of a boolean function f, denoted $D^{(k^+)}(f)$, is the minimum depth of any k^+ decision tree that computes it.

As usual, we also introduce *randomized k^+* decision trees. Formally, these are probability distributions $\mathcal{P}$ over deterministic k^+ decision trees. The *expected cost* of $\mathcal{P}$ on input x is $\mathbf{E}_{T \sim \mathcal{P}}[\text{cost}(T, x)]$. The expected cost of $\mathcal{P}$ itself is the maximum expected cost over all inputs x. Given a boolean function $f : \{0,1\}^n \to \{0,1\}$, the *error* of $\mathcal{P}$ on input x is $\mathbf{Pr}_{T \sim \mathcal{P}}[T(x) \neq f(x)]$. We say that $\mathcal{P}$ computes f with *zero error* if this error is 0 for all inputs x (in particular, each deterministic T in the support of $\mathcal{P}$ must compute f). We say that $\mathcal{P}$ computes f with *two-sided error* if the error is at most $1/3$ for all inputs x. Note that both the expected cost measure and the error measure are worst-case over all inputs; we do not consider distributional complexity in this article. The *zero (respectively, two-sided) error randomized k^+ decision tree complexity* of a boolean function f, denoted $R_0^{(k^+)}(f)$ (respectively, $R_2^{(k^+)}(f)$), is the minimum expected cost over all distributions $\mathcal{P}$ which compute f with zero (respectively, two-sided) error. In

this work, our randomized upper bounds will be for zero error k^+ computation and our randomized lower bounds for two-sided error.

We conclude by noting that the *simple* decision tree model is the 1^+ model with the extra restriction that the query sets Q_{pos} and Q_{neg} satisfy $|Q_{\text{pos}} \cup Q_{\text{neg}}| = 1$ at each node. We use the standard notation $D(f)$, $R_0(f)$, and $R_2(f)$ for the associated deterministic, 0-error, and 2-sided error complexities.

2.2 Related work

Our main results concern the complexity of symmetric functions under the 2^+ decision tree model, as well as the relation between the deterministic, zero-error randomized, and two-sided error randomized complexities of functions under the k^+ decision tree model. In this section, we review some of the work done on similar problems in the simple and 1^+ decision tree models.

Simple decision trees. The computation of totally symmetric functions is not interesting in the simple decision tree model; it's easy to see that for nonconstant totally symmetric f we have $D(f) = n$ (and it's also known that even $R_2(f) \geq \Omega(n)$ [16]). But some of the most interesting open problems in the theory of simple decision trees concern functions that have a large "degree" of symmetry. Recall that a *graph property* for v-vertex graphs is a decision problem $f : \{0,1\}^{\binom{v}{2}} \to \{0,1\}$ which is invariant under all permutations of the vertices. Let f be a nonconstant *monotone* graph property. Two famous open problem in simple decision tree complexity are the evasiveness conjecture [18], that $D(f)$ equals $\binom{v}{2}$, and the Yao-Karp conjecture [19], that $R(f)$ must be $\Omega(v^2)$.

The relationship between deterministic and randomized complexity is another interesting aspect of simple decision trees. Perhaps surprisingly, it is known [16] that deterministic, zero-error randomized, and two-sided error randomized simple decision tree complexity are polynomially related for every boolean function; specifically, $D(f) \leq O(R_2(f)^3)$. On the other hand, it is not known whether $R_0(f) \leq O(R_2(f))$ holds for all f.

1^+ decision trees. In the 1^+ model, the complexity of symmetric functions becomes a natural question, and a non-trivial one. For example, we of course have $D^{(1^+)}(T_n^1) = 1$, but the value of even $D^{(1^+)}(T_n^2)$ is not immediately obvious. Ben-Asher and Newman point out that it is not hard to show that $D^{(1^+)}(T_n^t) \leq O(t \log(n/t))$, and they show that this bound is tight:

Ben-Asher–Newman Theorem [3]. For $2 \leq t \leq n$, $D^{(1^+)}(T_n^t) = \Theta(t \log(n/t))$.

Ben-Asher and Newman also consider randomized complexity. They sketch the proof of the fact that $R_0^{(1^+)}(T_n^2) \geq \Omega(\log n)$, and also observe that $R_2^{(1^+)}(T_n^2) = O(1)$. This leads to the interesting conclusion that unlike in the simple decision tree model, there is *no* polynomial relationship between $R_0^{(1^+)}$ and $R_2^{(1^+)}$ — indeed, $R_0^{(1^+)}(f)$ can be *unboundedly* larger than $R_2^{(1^+)}(f)$.

As we mentioned in the introduction, Ben-Asher and Newman leave the study of the 2^+ model as an open problem. In particular, they asked if their main theorem can be extended to $D^{(2^+)}(T_n^t) \geq \Omega(t \log(n/t))$, observing only a trivial $\Omega(t)$ lower bound.

3 Our results

Our main results exactly characterize (up to constants) the zero and two-sided error randomized 2^+ complexities of all symmetric functions. We also nearly characterize the deterministic 2^+ complexity of symmetric functions; in particular, we answer the open question of Ben-Asher and Newman up to a $\log t$ factor.

Theorem 1. *For any symmetric boolean function $f : \{0,1\}^n \to \{0,1\}$, write $\Gamma = \Gamma(f)$. Then*

$$\Omega\big((\Gamma/\log\Gamma)\cdot\log(n/\Gamma)\big) \leq D^{(2^+)}(f) \leq O\big(\Gamma\cdot\log(n/\Gamma)\big),$$

$$R_0^{(2^+)}(f) = \Theta(\Gamma),$$

$$R_2^{(2^+)}(f) = \Theta(\Gamma).$$

In particular, the above bounds hold with $\Gamma = \min(t, n+1-t)$ for threshold functions $f = T_n^t$.

To get a feel for the threorem consider the threshold function $g = T_n^{100}$. Theorem 1 states that $R_0^{(2^+)}(g) = R_2^{(2^+)}(g) = O(1)$ while it can be shown that $R_0^{(2^+)}(g) \geq \Omega(\log n)$. On the other hand both $D^{(2^+)}(g) = D^{(1^+)}(g) = \Theta(\log n)$. The upper bounds and lower bounds of Theorem 1 are tackled in Sections 4 and 5 respectively.

An immediate corollary of Theorem 1 is that there is no polynomial relationship between deterministic and zero-error randomized 2^+ decision tree complexity; indeed, no bounded relationship at all. This is because for $t = O(1)$ we have $D^{(2^+)}(T_n^t) \geq \Omega(\log n)$, yet $R_0^{(2^+)}(T_n^t) = O(1)$. This latter result shows that the zero-error 2^+ decision tree model is quite powerful, being able to compute $T_n^{O(1)}$ with a number of queries *independent* of n.

Our upper bound $R_0^{(2^+)}(f) \leq O(\Gamma)$ relies essentially on the upper bound $R_0^{(2^+)}(T_n^t) \leq O(t)$, and to prove this we actually prove a stronger statement: any "k^+ query" can be exactly simulated with an expected $O(k)$ many 2^+ queries. Consequently we deduce that the zero error randomized k^+ decision tree complexity of *any* boolean function is $O(k)$ times smaller than its 2^+ decision tree complexity.

Corollary 1. *For any boolean function $f : \{0,1\}^n \to \{0,1\}$, $R_0^{(k^+)}(f) \geq \Omega(R_0^{(2^+)}(f)/k)$.*

The inequality in this corollary is best possible. Indeed, we show that for every symmetric function f it holds that $R_0^{(k^+)}(f) = \Theta(\Gamma(f)/k)$. A similar reduction can be made regarding deterministic k^+ complexity.

The full version of this article includes many other results regarding the k^+ decision tree complexity of general functions. We give a brief overview of some of these results in Section 6.

4 Upper bounds

A straightforward observation reduces computation of general symmetric functions to that of threshold functions. Thus in this section we only discuss computing threshold functions.

Our upper bound for the deterministic 2^+ complexity of thresholds follows immediately from the Ben-Asher–Newman Theorem (which in fact only needs 1^+ queries). We also have the following very straightforward extension:

Proposition 1. *Let $1 \leq k \leq t \leq n$ and $0 \leq t \leq n+1$ be integers. Then there exists a query algorithm that correctly decides whether $|x|$ is $0, 1, \ldots, t-1$, or at least t in $O(t/k \log(n/t))$ queries. In particular, $D^{(k^+)}(T_n^t) \leq \lceil \frac{t}{k} \rceil \cdot \left(2 \cdot \lceil \log\left(\frac{n}{t}\right) \rceil + 1\right)$.*

Note that this Proposition bounds the 2^+ complexity of a k^+ query, which immediately implies for every f that $D^{(k^+)}(f)$ is never better than $D^{(2^+)}(f)$ by more than a $O(k \log(n/k))$ factor.

To complete the proof of the upper bounds in Theorem 1, it now suffices to analyze the zero-error randomized complexity of threshold functions, which we do in the following theorem:

Theorem 2. *Let $2 \leq k \leq t \leq n$ be integers. Then there is a randomized k^+ query algorithm which, given $x \in \{0,1\}^n$, correctly decides whether $|x|$ is $0, 1, \ldots, t-1$, or at least t, using an expected $O(t/k)$ queries. In particular, $R_0^{(k^+)}(T_n^t) \leq O(t/k)$. Moreover, if $t \leq k$ then $R_0^{(k^+)}(T_n^t) = 1$.*

Corollary 1 follows directly from this theorem (using linearity of expectation), and as mentioned earlier Theorem 2 implies the upper bounds in Theorem 1.

The key to proving Theorem 2 is the following "COUNT" algorithm:

Theorem 3. *Let $k \geq 2$. There is an algorithm* COUNT *which on input x, outputs $|x|$ in an expected $O(1 + |x|/k)$ many k^+ queries.*

We will also need the following easier result:

Proposition 2. *For each $k \geq 1$ and each real t satisfying $k \leq t \leq n$, there is an $O(t/k)$-query zero-error randomized k^+ query algorithm which on input $x \in \{0,1\}^n$ has the following properties:*

(i) It makes (at most) $4t/k$ queries, with probability 1.
(ii) It outputs either "$|x| \geq t$" or "don't know."

(iii) If $|x| \geq 4t$, it outputs "$|x| \geq t$" with probability at least $1/4$.
(iv) If it ever outputs "$|x| \geq t$", then indeed $|x| \geq t$.

We remark that Proposition 2 works even if our k^+ queries only return the response "k^+" or "$< k$"; in particular, it holds even when $k = 1$. Theorem 3, however, needs the full power of k^+ queries.

The algorithm in Theorem 2 is as follows:

1. Run COUNT till $4t/k$ queries are made or it halts. In the latter case return the output of COUNT and halt.
2. Run algorithm from Proposition 2.
   a. If the algorithm outputs "don't know" go to Step 1 else output "$|x| \geq t$" and halt.

Next, we sketch how to prove Theorem 3 using balls and bins analysis. The COUNT algorithm involves randomly partitioning the coordinates $[n]$ into some number of "bins." We think of the "balls" as being the indices for which the input x has a 1. With this framework set up, it is fairly easy to prove Proposition 2. We also prove Theorem 3 about the COUNT algorithm using the balls and bins framework. Suppose we toss the balls (1-coordinates) into bins and then make a k^+ query on each bin. Recall that this tells us whether the number of balls is 0, 1, 2, ..., $k-1$, or $\geq k$. If a bin contains fewer than k balls, we say it *isolates* these balls. Whenever a bin isolates balls, we have made progress: we know exactly how many 1's are in x in the bin's coordinates. We can henceforth "throw away" these coordinates, remembering only the 1-count in them, and continue processing x on the substring corresponding to those coordinates in bins with at least k many 1's. Thus in terms of balls and bins, we can think of the task of counting $|x|$ as the task of isolating all of the balls. We note that the ability to isolate/count and throw away is the crucial tool that 2^+ queries gain us over 1^+ queries.

We now give a brief intuition behind the COUNT algorithm. Although the algorithm doesn't actually know $|x|$, the number of balls, if it could partition using $2|x|/k$ bins then that would likely isolate a constant fraction of the balls. If we could do this repeatedly while only using $O(\#\text{ balls remaining}/k)$ many queries, we will be able to construct the desired COUNT algorithm. Since we don't know the number of balls remaining, we can try using 2, 4, 8, etc., many bins. If we get up to around the "correct number" $2|x|/k$, we're likely to isolate a good fraction of balls; we can then reset back to 2 bins and repeat. Although we pay a query for each bin, we don't have to worry too much about doubling the number of bins too far; resetting becomes highly likely once the number of bins is at least the correct number. More worrisome is the possibility of resetting too early; we don't want to just keep isolating too few bins. However, we will show that if the number of bins is too small, we are very unlikely to get many isolating bins; hence we *won't* reset.

Statement of COUNT algorithm. We now present some more details on the COUNT algorithm. The COUNT algorithm is dovetailing of two other algorithms:

A-COUNT (which in $O(A^2)$ queries can determine if $|x| \leq A$, where A is an absolute constant) and SHAVE (to be described soon). More precisely,

0. Set $X \leftarrow 0$.
1. Run A-COUNT till A^2 queries are made. If the algorithm halts with the answer w then halt and return $X + w$.
2. Run SHAVE till A^2 queries are made. Assume SHAVE has isolated the set of indices S (out of which w are 1s), update $X \leftarrow X + w$ and go to Step 1 with the input projected to indices outside of S.

The algorithm SHAVE is as follows:

Run PARTITION$^+(t)$ with t equal to $0;\ 0,1;\ 0,1,2;\ 0,1,2,3;\ \ldots$.
Halt as soon as one of the run halts.

The algorithm PARTITION$^+(t)$ runs another algorithm PARTITION(t) 50 times and halts the first time PARTITION(t) "accepts" and "rejects" if all the runs "reject." Finally, the PARTITION(t) algorithm is as follows:

1. Toss the indices into 2^t bins and do a k^+ query on each bin.
2. Call a bin "good" if the number of balls in it is in the range $[\frac{1}{4}k, \frac{3}{4}k]$.
3. If the fraction of good bins is at least $\frac{1}{20}$, declare "accept" and isolate the balls.
4. Otherwise declare "reject" and *do not* isolate any balls.[a]

[a] Not isolating any ball is just for the simplicity of analysis. In practice one should indeed isolate any ball that one can.

5 Lower bounds

Deterministic lower bound.

Lemma 1. *For any symmetric function $f : \{0,1\}^n \to \{0,1\}$ such that $\Gamma = \Gamma(f) > 2$,*

$$D^{(2^+)}(f) \geq \Omega\left((\Gamma/\log\Gamma) \cdot \log(n/\Gamma)\right).$$

(When $\Gamma \leq 2$, $D^{(2^+)}(f) \leq 1$.)

Proof. The statement about the $\Gamma \leq 2$ case is trivial. For $\Gamma > 2$, assume without loss of generality that $f_{\Gamma-1} \neq f_\Gamma$. (If the inequality does not hold, then $f_{n-\Gamma} \neq f_{n-\Gamma+1}$ and we exchange the roles of the 0 and 1 labels in the rest of the proof.) Assume also for now that $\Gamma/3$ and $3n/\Gamma$ are integers. We describe an adversary that constructs inputs of weight $\Gamma - 1$ or Γ while answering the 2^+ queries of the algorithm consistently.

The adversary maintains two pieces of information: a list of $m = \Gamma/3$ sets $U_1, \ldots, U_m$ of "undefined" variables and a set $I \subseteq [m]$ of the sets of undefined variables that are "active." Initially, $U_\ell = \{\frac{n}{m}(\ell-1)+1, \ldots, \frac{n}{m} \cdot \ell\}$ and $I = [m]$. For each query $(Q_{\text{pos}}, Q_{\text{neg}})$, the adversary proceeds as follows:

1. If there is an index $\ell \in I$ such that $|Q_{\text{pos}} \cap U_\ell| > |U_\ell|/m$, then the adversary answers "2^+", assigns the variables in $U_\ell \setminus Q_{\text{pos}}$ the value 0, and updates $U_\ell = U_\ell \cap Q_{\text{pos}}$. We refer to a query handled in this manner as an ℓ-query.
2. Otherwise, let $Q' \subseteq Q_{\text{neg}}$ be a set of size $|Q'| = \min\{2, |Q_{\text{neg}}|\}$. The adversary sets the variables in $U_\ell \cap (Q_{\text{pos}} \cup Q')$ to 0 and updates $U_\ell = U_\ell \setminus (Q_{\text{pos}} \cup Q')$ for each $\ell \in I$. It then returns the answer "0", "1", or "2^+", depending on the size of Q'. We refer to the query as a 0-query in this case.

After answering the query, each set U_ℓ of size $|U_\ell| < 3m$ is considered "defined." When the set U_ℓ is defined, the adversary updates $I = I \setminus \{\ell\}$. If I is still not empty, the adversary also sets 3 of the variables in U_ℓ to one. When the last set U_ℓ is defined, the adversary sets either 2 or 3 of its variables to one.

While not all the sets are defined, the answers of the adversary are consistent with inputs of weight $\Gamma - 1$ and Γ. Therefore, the algorithm must make enough queries to reduce the sizes of $U_1, \ldots, U_m$ to less than $3m$ each. Let $q = q_0 + q_1 + \cdots + q_m$ be the number of queries made by the algorithm, where q_0 represents the number of 0-queries and q_ℓ represents the number of ℓ queries, for $\ell = 1, \ldots, m$.

Consider now a fixed $\ell \in [m]$. Each ℓ-query removes at most a $1-1/m$ fraction of the elements in U_ℓ, and each 0-query removes at most $|U_\ell|/m + 2 \leq 2|U_\ell|/m$ elements. So $|U_\ell| < 3m$ holds only when

$$\left(\frac{n}{m}\right)\left(\frac{1}{m}\right)^{q_\ell}\left(1-\frac{2}{m}\right)^{q_0} < 3m.$$

The inequality holds for each of $\ell = 1, \ldots, m$; taking the product of the m inequalities, we obtain $\left(\frac{n}{m}\right)^m \left(\frac{1}{m}\right)^{q_1+\cdots+q_m}\left(1-\frac{2}{m}\right)^{m\cdot q_0} < (3m)^m$, which implies (for $m \geq 4$)

$$\left(\frac{n}{3m^2}\right)^m < m^{q_1+\cdots+q_m}\left(1-\frac{2}{m}\right)^{-m\cdot q_0} \leq m^{q_1+\cdots+q_m} 4^{2q_0} \leq m^{2(q_0+q_1+\cdots+q_m)}.$$

Taking the logarithm on both sides and dividing by $2\log m$, we get $\frac{m}{2\log m}\log\frac{n}{3m^2} < q_0 + q_1 + \cdots + q_m = q$. Recalling that $m = \Gamma/3$, we get the desired lower bound.

To complete the proof, we now consider the case where $\Gamma/3$ or n/m is not an integer. In this case, let Γ' be the largest multiple of 3 that is no greater than Γ, let $n' = n - (\Gamma - \Gamma')$, and let n'' be the largest multiple of m no greater than n'. Let the adversary fix the value of the last $n - n'$ variables to one and the previous $n' - n''$ variables to zero. We can now repeat the above argument with Γ' and n'' replacing Γ and n.

Randomized lower bound. We have the following result:

Lemma 2. *For any symmetric function $f : \{0,1\}^n \to \{0,1\}$ such that $\Gamma = \Gamma(f) > k$ and integer $k \geq \Gamma$,*

$$R_2^{(k^+)}(f) \geq \Omega\left(\Gamma/k\right).$$

When $\Gamma \leq k$, even $D^{(2^+)}f \leq 1$.

6 Other results

In the full version of this article, we prove a number of results about the deterministic k^+ complexity of non-symmetric functions. We state a number of these results in this section.

First, every boolean function has nontrivial deterministic k^+ complexity:

Theorem 4. *For all* $f : \{0,1\}^n \to \{0,1\}$, $D^{(k^+)}(f) \leq O(n/\log k)$.

The proof of this theorem uses a result from the combinatorial group testing literature [10]. The bound in Theorem 4 is sharp:

Theorem 5. *At least a* $1 - 2^{-2^{n-1}}$ *fraction of functions* $f : \{0,1\}^n \to \{0,1\}$ *satisfy the inequality* $D^{(k^+)}(f) \geq (n/\log(k+1))(1 - o_n(1))$.

We can furthermore exhibit simple explicit functions with this property:

Theorem 6. *Let* $\textsc{eq} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *be defined by* $\textsc{eq}(x,y) = 1$ *iff* $x = y$. *Then* $D^{(k^+)}(\textsc{eq}) \geq \Omega(n/\log k)$.

The proof of Theorem 6 is a direct corollary of a more general result linking deterministic k^+ complexity to communication complexity. Let $\mathrm{CC}(f)$ denote the deterministic 2-party communication complexity of f (for details, see [14]).

Theorem 7. *For any* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, $D^{(k^+)}(f) \geq \Omega(\mathrm{CC}(f)/\log k)$.

Interestingly, the deterministic k^+ complexity of a function is also closely related to its simple decision tree *rank*. The notion of (simple) decision tree rank was first introduced by Ehrenfeucht and Haussler [12] in the context of learning theory, and has the following recursive definition. If T has a single (leaf) node we define $\textsc{rank}(T) = 0$. Otherwise, supposing the two subtrees of T's root node are T_1 and T_2, we define $\textsc{rank}(T) = \max\{\textsc{rank}(T_1), \textsc{rank}(T_2)\}$ if $\textsc{rank}(T_1) \neq \textsc{rank}(T_2)$, and $\textsc{rank}(T) = \textsc{rank}(T_1) + 1$ if $\textsc{rank}(T_1) = \textsc{rank}(T_2)$. For a boolean function f, we define $\textsc{rank}(f)$ to be the minimum rank among simple decision trees computing f.

Theorem 8. *For all* $f : \{0,1\}^n \to \{0,1\}$,

$$\textsc{rank}(f)/k \leq D^{(k^+)}(f) \leq O\big(\textsc{rank}(f) \log(n/\textsc{rank}(f))\big).$$

Both bounds in this inequality may be tight. For the lower bound, it can be shown that for any symmetric function f we have $\textsc{rank}(f) = \Theta(\Gamma(f))$. This implies that for $t = \Theta(n)$ we have $\textsc{rank}(T_n^t) = \Theta(n)$; but for this t we also have $D^{(k^+)}(T_n^t) \leq O(n/k)$, by Proposition 1. This does not rule out a lower bound of the form $(\textsc{rank}(f)/k) \cdot \log(n/\textsc{rank}(f))$, but such a lower bound would be ruled out by the OR function, which has rank 1 but even 1^+ query complexity 1. The upper bound is tight in the case of the so-called ODD-MAX-BIT function, which has rank 1; it can be shown that $D^{(k^+)}(\textsc{odd-max-bit}) \geq \Omega(\log n)$, *independent* of k.

Finally, in contrast to the evasiveness conjecture (for simple decision trees), we show that the basic monotone graph property of *connectivity* has $o(v^2)$ deterministic 1^+ complexity:

Theorem 9. *For the connectivity graph property* $\textsc{Conn}_v : \{0,1\}^{\binom{v}{2}} \to \{0,1\}$ *it holds that* $D^{(1^+)}(\textsc{Conn}_v) \leq v(\lceil \log v \rceil + 1)$.

References

1. Aigner, M.: Combinatorial Search. Wiley-Teubner Series in Computer Science (1988)
2. Arora, A., Dutta, P., Bapat, S., Kulathumani, V., Zhang, H., Naik, V., Mittal, V., Cao, H., Demirbas, M., Gouda, M., Choi, Y.R., Herman, T., Kulkarni, S.S., Arumugam, U., Nesterenko, M., Vora, A., Miyashita, M.: A line in the sand: A wireless sensor network for target detection, classification, and tracking. Computer Networks (Elsevier) 46(5), 605–634 (2004)
3. Ben-Asher, Y., Newman, I.: Decision trees with boolean threshold queries. J. Comput. Syst. Sci. 51(3), 495–502 (1995)
4. Ben-Or, M.: Lower bounds for algebraic computation trees. In: STOC '83. pp. 80–86 (1983)
5. Bshouty, N.H.: A subexponential exact learning algorithm for DNF using equivalence queries. Information Processing Letters" 59(3), 37–39 (1996)
6. Buhrman, H., de Wolf, R.: Complexity measures and decision tree complexity: A survey. Theoretical Computer Science 288(1), 21–43 (2002)
7. Chockler, G., Demirbas, M., Gilbert, S., Lynch, N.A., Newport, C.C., Nolte, T.: Consensus and collision detectors in radio networks. Distributed Computing 21(1), 55–84 (2008)
8. Demirbas, M., Soysal, O., Hussain, M.: Singlehop collaborative feedback primitives for wireless sensor networks. INFOCOM pp. 2047–2055 (2008)
9. Dobkin, D., Lipton, R.J.: Multidimensional searching problems. SIAM Journal on Computing 5(2), 181–186 (1976), http://link.aip.org/link/?SMJ/5/181/1
10. Du, D.Z., Hwang, F.K.: Combinatorial Group Testing and its Applications. World Scientific (2000)
11. Dutta, P., Musaloiu-e, R., Stoica, I., Terzis, A.: Wireless ack collisions not considered harmful. In: In HotNets-VII: The Seventh Workshop on Hot Topics in Networks (2008)
12. Ehrenfeucht, A., Haussler, D.: Learning decision trees from random examples. Information and Computation 82(3), 231–246 (1989)
13. Kushilevitz, E., Mansour, Y.: Learning decision trees using the fourier spectrum. SIAM Journal on Computing 22(6), 1331–1348 (1993), http://link.aip.org/link/?SMJ/22/1331/1
14. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press (1997)
15. Moran, S., Snir, M., Manber, U.: Applications of ramsey's theorem to decision tree complexity. J. ACM 32(4), 938–949 (1985)
16. Nisan, N.: CREW PRAMs and decision trees. SIAM Journal on Computing 20(6), 999–1007 (1991), http://link.aip.org/link/?SMJ/20/999/1
17. Paturi, R.: On the degree of polynomials that approximate symmetric boolean functions (preliminary version). In: STOC '92. pp. 468–474 (1992)
18. Rosenberg, A.L.: On the time required to recognize properties of graphs: a problem. SIGACT News 5(4), 15–16 (1973)
19. Yao, A.C.C.: Monotone bipartite graph properties are evasive. SIAM Journal on Computing 17(3), 517–520 (1988)