On the Goal Value of a Boolean Function

Eric Bach
Dept. of CS
University of Wisconsin
1210 W. Dayton St.
Madison, WI 53706

Lisa Hellerstein
Dept of CSE
NYU School of Engineering
2 Metrotech Center, 10th Floor
Brooklyn, NY 11201

Devorah Kletenik
Dept. of CIS
Brooklyn College
City University of New York
2900 Bedford Avenue
Brooklyn, NY 11210

Abstract

In recent work, Deshpande et al. introduced a new measure of the complexity of a Boolean function (Deshpande, Hellerstein, and Kletenik 2014). The measure is related to the maximum utility value of a certain monotone, submodular utility function associated with the Boolean function. We call this measure the "goal value" of the function. Proving that a Boolean function has small goal value can yield new bounds on the average-depth decision tree complexity of the function, and new approximation algorithms for its sequential testing problem. We present bounds on the goal values of arbitrary and specific Boolean functions. We also compare the goal value measure to other, previously studied, measures of the complexity of Boolean functions.

Introduction

In recent work, Deshpande et al. introduced a new measure of the complexity of a Boolean function (Deshpande, Hellerstein, and Kletenik 2014). We call this measure the "goal value" of the function.[1] The measure is related to the maximum utility value of a certain monotone, submodular utility function associated with the Boolean function.

Proving that a Boolean function f has small goal value can lead to a good approximation algorithm for the Stochastic Boolean Function Evaluation problem for f. Also, if f has small goal value, it indicates a close relationship between two other measures of the complexity of f, its average-case decision tree complexity and its average-case certificate complexity.

In this work, we explore properties of the goal value measure. We review previous results on the measure. We prove new bounds on the goal values of general and specific Boolean functions. We also consider the related 1-goal value and 0-goal value measures. Finally, for monotone Boolean functions, we present a result relating goal value to other, previously studied, measures of Boolean function complexity.

We begin by presenting necessary definitions and explaining the known connections between goal value, decision tree complexity, and Boolean function evaluation. We then show that the goal value of every Boolean function f is at least n, if f depends on all n of its input variables. We note that this lower bound is tight for certain Boolean functions, such as AND, OR and XOR functions. We also present goal value bounds for Boolean k-of-n functions.

Deshpande et al. showed that the goal value of every Boolean function f is at most $\mathrm{ds}(f) \cdot \mathrm{cs}(f)$, where $\mathrm{ds}(f)$ is the minimum number of terms in a DNF formula for the function, and $\mathrm{cs}(f)$ is the minimum number of clauses in a CNF formula for the function.

It is well known and easy to show that $ds(f) \leq 2^{n-1}$ and $cs(f) \leq 2^{n-1}$. Thus the goal value of f is upper-bounded by 2^{2n-2}. We do not know how close goal value can come to this upper bound, but we show that there is a function whose goal value is approximately $2^{\frac{n}{2}}$. We also show that the goal value of a read-once function is at least the maximum of $\mathrm{cs}(f)$ and $\mathrm{ds}(f)$. Note that there is only a quadratic gap between this lower bound and the upper bound of $\mathrm{ds}(f) \cdot \mathrm{cs}(f)$.

We show that the polynomial threshold degree of a monotone Boolean function is a lower bound on its goal value. It can be a very weak lower bound, however, because threshold degree is at most n, whereas goal value can be exponential in n.

Terminology and Notation

Boolean function definitions: Let $V = \{x_1, \ldots, x_n\}$. A partial assignment is a vector $b \in \{0, 1, *\}^n$. For partial assignments $a, b \in \{0, 1, *\}^n$, a is an *extension* of b, written $a \succeq b$, if $a_i = b_i$ for all $b_i \neq *$. We also say that b is *contained* in a.

Given Boolean function $f : \{0, 1\}^n \to \{0, 1\}$, a partial assignment $b \in \{0, 1, *\}^n$ is a *0-certificate* of f if $f(a) = 0$ for all $a \in \{0, 1\}^n$ such that $a \succeq b$. It is a *1-certificate* if $f(a) = 1$ for all $a \in \{0, 1\}^n$ such that $a \succeq b$. It is a *certificate* if it is either a 0-certificate or a 1-certificate. We say that b *contains* a variable x_i if $b_i \neq *$. We will occasionally treat a certificate b as being the set of variables x_i such that $b_i \neq *$, together with the assignments given to them by b. The *size* of a certificate b is the number of variables x_i that it contains. If b and b' are certificates for x with $b \succeq b'$, we say that b' *is contained in* b. A certificate b for f is *minimal* if there is no certificate b' of f, such that $b' \neq b$ and $b \succeq b'$.

The *certificate size* of a Boolean function f is the maximum, over all $x \in \{0, 1\}^n$, of the size of the smallest certificate contained in x. Given a distribution D on $\{0, 1\}^n$,

[1] Deshpande et al. called it the *Q-value*. Since Q has no particular meaning, we have chosen to use a more intuitive name.

the *expected certificate size* of f is the expected value, for x drawn from D, of the size of the smallest certificate contained in x.

It is well-known that every Boolean function f can be represented as a *polynomial over GF(2)*. That is, either f is the constant function $f = 0$, or f is computed by an expression of the form $T_1 + T_2 + \ldots + T_k$ for some $k > 0$, where each T_i is the conjunction (i.e., AND) of the variables in some subset of V, no two terms contain exactly the same subset S_i of variables, and addition is modulo 2. (If $S_i = \emptyset$, then $T_i = 1$.) This is sometimes called the ring-sum expansion of the function. Each Boolean function has a unique such representation, up to permutation of the terms T_i. A variable x_i appears in the GF(2) polynomial for Boolean function f iff function f depends on variable x_i.

For function $f(x_1, \ldots, x_n)$ defined on $\{0,1\}^n$, and $b \in \{0, 1, *\}$, the function f_b *induced on* f *by* b is defined as follows. Let $V_b = \{x_i | b_i = *\}$. Then V_b is the set of input variables for function f_b, and the domain of f_b is the set of assignments that assign 0 or 1 to each variable. For each a in the domain, $f_b(a) = f(a \backslash b)$, where $a \backslash b$ denotes the assignment to V produced by setting the variables in V_b according to a, and the variables in $V - V_b$ according to b. For $k \in \{0, 1\}$, we use $f_{x_i \leftarrow k}$ to denote the function induced on f by the partial assignment setting x_i to k, and all other variables to $*$.

A k-of-n function is a Boolean function on n variables whose output is 1 if and only if at least k of its input variables are 1.

A *read-once formula* is a monotone Boolean formula over the basis of AND and OR gates, whose inputs are all variables. We can view a read-once formula as a rooted tree with the following properties:

1. The leaves are labeled with distinct variables from the set $\{x_1, \ldots, x_n\}$, for some $n \geq 1$
2. Each internal node is labeled AND or OR
3. Each node labeled AND or OR gate has at least 2 children.

A *maxterm* of a monotone Boolean function $f(x_1, \ldots, x_n)$ is the set of variables contained in a minimal 0-certificate of f. (Equivalently, it is a set of variables S such that setting the variables in S to 0 forces f to 0, but this is not true of any proper subset of S.) A *minterm* of f is the set of variables contained in a minimal 1-certificate of f. If F is a Boolean formula expressing a monotone function, we say that a subset S of its variables is a maxterm (minterm) of F iff S is a maxterm (minterm) of the function is computes. The *canonical DNF formula* for a non-constant monotone Boolean function f is a formula of the form $T_1 \vee T_2 \vee \ldots \vee T_k$ where k is the number of minterms of f, and each T_i is the conjunction (AND) of the variables in a distinct minterm of f. It is unique up to permutation of terms.

A decision tree computing a Boolean function $f(x_1, \ldots, x_n)$ is a binary tree where each binary node is labeled by a variable in V, and each each leaf is labeled either 0 or 1. The left child of an internal node labeled x_i corresponds to $x_i = 0$, and the right child corresponds to $x_i = 1$. The tree computes f if for each $x \in \{0,1\}^n$, the root-leaf path induced by x ends in a leaf whose label is $f(x)$. Given a decision tree T computing f, and $x \in \{0,1\}^n$, let $Depth(T, x)$ denote the number of internal nodes on the root-leaf path in T induced by x. Then the depth of T is the maximum value of $Depth(T, x)$ for all $x \in \{0,1\}^n$. Given a probability distribution on $\{0,1\}^n$, the *expected depth of* T is the expected value of $Depth(T, x)$, when x is drawn from the distribution. The *average depth* of T is its expected depth with respect to the uniform distribution on $\{0,1\}^n$. The *decision tree depth* of Boolean function f is the minimum depth of any decision tree computing f. The *expected decision tree depth* of Boolean function f, with respect to a distribution on $\{0,1\}^n$, is the minimum expected depth of any decision tree computing f, with respect to that distribution.

The *rank* of a (binary) decision tree T is defined as follows (Ehrenfeucht and Haussler 1989): if T is a leaf, then $\texttt{rank}(T) = 0$. Otherwise, let $\texttt{rank}_0$ denote the rank of the subtree rooted at the left child of the root of T and $\texttt{rank}_1$ denote the rank of the subtree rooted at the right child of the root of T. Then

$$\texttt{rank}(T) = \begin{cases} \max\{\texttt{rank}_0, \texttt{rank}_1\} & \text{if } \texttt{rank}_0 \neq \texttt{rank}_1 \\ \texttt{rank}_0 + 1 & \text{otherwise} \end{cases}$$

A *k-decision list* is a list of pairs (t_i, v_i) where t_i is a term (conjunction) of at most k literals and $v_i \in \{0, 1\}$. The last term t_i is the empty term, which is equal to 1. A decision list L defines a Boolean function such that for any assignment $x \in \{0,1\}^n$, $L(x) = v_j$ where j is the minimum index such that $t_j(x) = 1$ (Rivest 1987).

We say that a Boolean function f is computed by a *degree d polynomial threshold function* if there is a multivariate polynomial $p(x_1, \ldots, x_n)$ of degree d, over the real numbers, such that for all $x \in \{0,1\}^n$, $f(x) = \text{sgn}(p(x))$, where $\text{sgn}(z) = 1$ if $z \geq 0$ and $\text{sgn}(z) = 0$ if $z < 0$. The *polynomial threshold degree* of f is the minimum d such that f is computed by a polynomial threshold function of degree d.

Submodularity Definitions: Let $N = \{1, \ldots, n\}$. In what follows, we assume that utility functions are integer-valued. In the context of standard work on submodularity, a *utility function* is a set function $g : 2^N \rightarrow \mathbb{Z}_{\geq 0}$. Given $S \subseteq N$ and $j \in N$, $g_S(j)$ denotes the quantity $g(S \cup \{j\}) - g(S)$.

We also use the term *utility function* to refer to a function $g : \{0, 1, *\}^n \rightarrow \mathbb{Z}_{\geq 0}$ defined on partial assignments. For $l \in \{0, 1, *\}$, the quantity $b_{x_i \leftarrow l}$ denotes the partial assignment that is identical to b except that $b_i = l$.

A utility function $g : \{0, 1, *\}^n \rightarrow \mathbb{Z}_{\geq 0}$ is *monotone* if for $b \in \{0, 1, *\}^n$, $i \in N$ such that $b_i = *$, and $l \in \{0, 1\}$, $g(b_{x_i \leftarrow l}) - g(b) \geq 0$; in other words, additional information can not decrease utility. Utility function g is *submodular* if for all $b, b' \in \{0, 1, *\}^n$ and $l \in \{0, 1\}$, $g(b_{x_i \leftarrow l}) - g(b) \geq g(b'_{x_i \leftarrow l}) - g(b')$ whenever $b' \succeq b$ and $b_i = b'_i = *$. This is a diminishing returns property.

We say that $Q \geq 0$ is the *goal value* of $g : \{0, 1, *\}^n \rightarrow \mathbb{Z}_{\geq 0}$ if $g(b) = Q$ for all $b \in \{0, 1\}^n$.

A *goal function* for a Boolean function $f : \{0,1\}^n \rightarrow \{0, 1\}$ is a monotone, submodular function $g : \{0, 1, *\}^n \rightarrow$

$\mathbb{Z}_{\geq 0}$, with goal value Q, such that for all $b \in \{0,1,*\}^n$, $g(x) = Q$ iff b contains a certificate of f. We define the *goal value of* f to be the minimum goal value of any goal function g for f.

The goal value of f is clearly equal to the goal value of the complement of the function f.

A 1-goal function for f is a monotone, submodular function $g : \{0,1,*\}^n \to \mathbb{Z}_{\geq 0}$, such that for some constant $Q \geq 0$, $g(b) = Q$ if b is a 1-certificate of f, and $g(b) < Q$ otherwise. We call Q the 1-goal value of g.

We define the *1-goal value of* f to be the minimum goal value of any 1-goal function for f. The definition of a *0-goal function* for f and the *0-goal value* of f are analogous.

We denote the goal value, 1-goal value, and 0-goal value of f by $\Gamma(f)$, $\Gamma^1(f)$, and $\Gamma^0(f)$ respectively.

We use $g(x_i = \ell)$ to denote the value of $g(b)$ for $b \in \{0,1,*\}^n$ such that $b_i = \ell$, and $b_j = *$ for $j \neq i$.

Stochastic Boolean Function Evaluation

In the *Stochastic Boolean Function Evaluation* (SBFE) problem, we are given a representation of a Boolean function $f : \{0,1\}^n \to \{0,1\}$, where the representation is drawn from a fixed class of representations C. For example, C might be the class of read-once formulas. We are also given a vector of probabilities $(p_1, \ldots, p_n) \in (0,1)^n$, and a vector of costs $(c_1, \ldots, c_n) \in \mathbb{Z}^n_{\geq 0}$. We would like to determine the value of f on an initially unknown input $x \in \{0,1\}^n$, when cost c_i must be paid to get the value of x_i, and each bit x_i of x is equal to 1 with independent probability p_i. The problem is to find the best (adaptive) order in which to acquire the bits of x, so as to minimize the expected cost incurred before $f(x)$ can be determined. SBFE problems arise in diverse application areas, including medical diagnosis, database query optimization, Operations Research, and machine learning with attribute costs (Ibaraki and Kameda 1984; Krishnamurthy, Boral, and Zaniolo 1986; Deshpande and Hellerstein 2008; Srivastava et al. 2006; Ünlüyurt 2004; Kaplan, Kushilevitz, and Mansour 2005).

Deshpande et al. observed that an instance of the SBFE problem for a Boolean function f can be reduced to an instance of a problem called *Stochastic Submodular Set Cover* (SSSC), by constructing a goal function g for f (Deshpande, Hellerstein, and Kletenik 2014). Golovin and Krause gave an algorithm for the SSSC problem, called Adaptive Greedy, that achieves an $O(\log Q)$ approximation, when given as input a utility function $g : \{0,1,*\}^n \to \mathbb{Z}_{\geq 0}$ with goal value Q (Golovin and Krause 2011). Applying it to the goal function g constructed for Boolean function f yields an $O(\log Q)$ approximation for the SBFE problem for f.

Recall that $\mathrm{ds}(f)$ is the minimum number of terms in a DNF formula for f, and $\mathrm{cs}(f)$ is the minimum number of clauses in a CNF formula for f. Deshpande et al. gave a generic method for constructing a goal function g for an arbitrary Boolean function f, such that g has goal value $\mathrm{ds}(f) \cdot \mathrm{cs}(f)$. Hence, by running Adaptive Greedy on g, the SBFE problem for f can be approximated to within a factor of $O(\log(\mathrm{ds}(f) \cdot \mathrm{cs}(f)))$ when f is given by its CNF and DNF formulas (or by a Boolean decision tree). Deshpande et al. also showed that for certain Boolean functions, $\Gamma(f)$ can be exponential in n. In particular, they showed that $\Gamma(f) \geq 2^{n/2}$ both for the function $f(x_1, \ldots, x_n) = x_1x_2 \vee x_3x_4 \vee \ldots x_{n-1}x_n$, and for a certain linear-threshold function having coefficients of exponential magnitude. For such functions, the approach of constructing a goal function g for f, and then running Adaptive Greedy on g, does not yield a good approximation algorithm for the SBFE problem for f. See (Deshpande, Hellerstein, and Kletenik 2014) for further details.

Bounds on Decision Tree Depth

In (Deshpande, Hellerstein, and Kletenik 2013), it was shown that the goal value of a Boolean function f could be used to upper bound the expected depth of a decision tree computing f, with respect to the uniform distribution on $\{0,1\}^n$.

Let $\mathsf{DT}(f)$ denote the expected decision tree depth of f, and $\mathsf{CERT}(f)$ denote the expected certificate size of f, where both expectations are with respect to the uniform distribution on $\{0,1\}^n$. It is easy to show that $\mathsf{CERT}(f) \leq \mathsf{DT}(f)$. Let g be a goal function for f whose goal value is $\Gamma(f)$. Under the uniform distribution, when run on g, the Adaptive Greedy algorithm of Golovin and Krause outputs a solution (corresponding to a decision tree) that is within a factor of $2(\ln \Gamma(f) + 1)$ of the expected certificate size of f. It follows that

$$\mathsf{DT}(f) \leq (2 \ln \Gamma(f) + 1)\mathsf{CERT}(f).$$

Thus if $\Gamma(f)$ is small, expected certificate size and expected depth are close to each other. However, $\Gamma(f)$ can be very large. In (Allen et al. 2014), it was shown that the gap between $\mathsf{DT}(f)$ and $\mathsf{CERT}(f)$ can be exponential: there is a function f such that for any constant $0 < \epsilon < 1$, $\frac{\mathsf{DT}(f)}{\mathsf{CERT}(f)} = \Omega(2^{\epsilon \mathsf{CERT}(f)})$. This stands in marked contrast to results for *worst-case* depth of decision tree and worst-case certificate size; it is known that the worst-case depth of a decision tree is at most quadratic in the size of the worst-case certificate (Buhrman and Wolf 1999).

Lower Bound on Goal Value

In this section, we seek to lower bound the goal value for all Boolean functions. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function depending on all its variables.

We begin with the following lemma:

Lemma 1. *Let $f(x_1, \ldots, x_n)$ be a Boolean function. Let $\ell, k \in \{0,1\}$. Let g be a goal function for f. If there is a minimal certificate of f setting $x_i = \ell$, then $g(x_i = \ell) - g(*, \ldots, *) \geq 1$.*

Similarly, if g' is a k-goal function for f, and there is a minimal k-certificate for f setting $x_i = \ell$, which has size s, then $g'(x_i = \ell) - g'(*, \ldots, *) \geq 1$. Further, if the size of that certificate is s, then the goal value of g' is at least s.

Proof. Let g be a goal function for f, and let Q be its goal value. Let g' be a k-goal function for f, and let Q' be its k-goal value. Let d be a minimal k-certificate for f. Thus $g(d) = Q$ and $g'(d) = Q'$.

Let s be the size of certificate d, and without loss of generality, assume that $x_1, \ldots, x_s$ are the variables contained in d. Consider the sequence $d^0, d^1, \ldots, d^s$ where $d^0 = (*, \ldots, *)$ and $d^i = (d_1, d_2, \ldots, d_i, *, \ldots, *)$ for $1 \leq i \leq s$. Consider a fixed i such that $1 \leq i \leq s$. By monotonicity, $g(d^{i-1}) \leq g(d_i)$. Suppose $g(d^{i-1}) = g(d_i)$. Let $\hat{d}$ be the partial assignment that is obtained from d^s by setting x_i to $*$. Since d is a minimal certificate for f, $\hat{d}$ is not a certificate, and so $g(\hat{d}) < Q$, and thus $g(d) - g(\hat{d}) \geq 1$. Since d and $\hat{d}$ differ only in the assignment to x_i, and $\hat{d} \succeq d^{i-1}$, by submodularity, $g(d^i) - g(d^{i-1}) \geq 1$ and $g(x_i = d_i) - g(*, \ldots, *) \geq 1$.

The same argument shows that $g'(d^i) - g'(d^{i-1}) \geq 1$ and $g'(x_i = d_i) - g(*, \ldots, *) \geq 1$. From the first inequality, we get that $g'(d) \geq s$. The lemma follows. □

The proof of the above lemma also shows that $\Gamma(f) \geq d$, where d is the size of a minimal certificate for f. But $d \leq n$, and we will prove that $\Gamma(f) \geq n$. First, we prove the following lemma.

Lemma 2. *Let $f(x_1, \ldots, x_n)$ be a Boolean function depending on all n of its variables. Then there exists a variable x_i and a value $\ell \in \{0, 1\}$ such that the induced function $f_{x_i \leftarrow \ell}$ depends on all $n - 1$ of its input variables.*

Proof. Consider the GF(2) polynomial computing f. The polynomial must contain all n variables, since it depends on all of them.

For each x_i, let $T(x_i)$ denote the set of terms of the GF(2) polynomial which contain x_i. Define a partial order on those variables x_i such that $x_i < x_j$ in the partial order if $T(x_i)$ is a proper subset of $T(x_j)$.

Now consider an x_i such that $T(x_i)$ is a minimum element in this partial order. We analyze several different cases:

Case 1: The variable x_i appears in all non-constant terms of the polynomial. In this case, setting x_i to 1 produces a polynomial that contains the remaining $n - 1$ variables, and cannot be simplified (since it has no repeated terms). Thus the induced function must depend on all those variables.

Case 2: The variable x_i does not appear in all terms of the polynomial, and there is no $x_i \neq x_j$ such that $T(x_i) = T(x_j)$. In this case, setting x_i to 0 just deletes all terms of the polynomial containing x_i, and the result is the polynomial for the induced function that cannot be simplified. By assumption, for each $x_j \neq x_i$, $T(x_i) \neq T(x_j)$, and since $T(x_i)$ is a minimum, there must be a term containing x_j but not x_i. This term is in the polynomial that remains after setting x_i to 0, and so the function setting x_i to 0 depends on all remaining x_j.

Case 3: The variable x_i does not appear in all terms of the polyomial, and there exists an $x_j \neq x_i$ such that $T(x_i) = T(x_j)$. In this case, set x_i to 1 in the polynomial. The result does not have any repeated terms: if t was a term containing x_i in the original polynomial, then there cannot be a term in the polynomial containing exactly the variables in t that are not equal to x_i, because then it would contain x_j, implying that $T(x_i) \neq T(x_j)$. Thus the result of setting x_i to 1 is to produce a polynomial that is the same as the original polynomial, except that x_i is deleted from any terms containing it (and no simplification is possible). Since all of the variables appeared in the original polynomial, all of the variables except x_i appear in the resulting polynomial. □

We now have the following theorem:

Theorem 1. *Let $f : \{0, 1\}^n \to \{0, 1\}$ be a Boolean function. If f depends on exactly n' of its input variables, then $\Gamma(f) \geq n'$. Further, $\Gamma^1(f) + \Gamma^0(f) \geq n' + 1$.*

Proof. We prove the theorem in the case that f depends on all n of its input variables, so $n' = n$. This is without loss of generality, because if $n' < n$, we can consider instead the restriction of f to the variables on which it depends. It is easy to verify that this causes no change in the values of Γ, Γ^1, and Γ^0.

By repeated application of Lemma 2, it follows that there exists a sequence of partial assignments, $b^0, b^1, \ldots, b^{n'}$ in $\{0, 1, *\}^n$, such that b^0 is the all-$*$ assignment, b^n has no $*$'s, each b_{i+1} is an extension of b^i produced by changing exactly one $*$ to a non-$*$ value, and the function f^i that is induced from f by partial assignment b^i depends on all $n - i$ of its variables.

Without loss of generality, assume b^i assigns values in $\{0, 1\}$ to variables $x_{n-i+1}, \ldots, x_n$, and sets variables $x_1, \ldots, x_{n-i}$ to $*$. Thus $f^i : \{0, 1\}^{n-i} \to \{0, 1\}$.

Let g be a goal function for f. For $i \in \{1, \ldots, n\}$, let $g^i : \{0, 1, *\}^{n-i} \to \mathbb{Z}_{\geq 0}$ be the function induced on g by b^i. Thus for $a \in \{0, 1, *\}^{n-i}$, $g^i(a) = g(a \backslash b^i)$, where $a \backslash b^i$ denotes the assignment produced by setting the variables in $\{x_1, \ldots, x_{n-i}\}$ according to a, and the remaining variables of g according to b^i. From the fact that g is a goal function for f, it easily follows that g^i is a goal function for f^i, with goal value equal to the goal value of g.

Let $x_{n-i+1} \leftarrow \ell$, where $\ell \in \{0, 1\}$, be the one-variable assignment that extends b^{i-1} to produce b^i. Since f^{i-1} depends on x_{n-i+1}, there is an assignment $b \in \{0, 1\}^{n-i+1}$ such that $f^{i-1}(b_{x_{n-i+1}} \leftarrow 0) \neq f^{i-1}(b_{x_{n-i+1}} \leftarrow 1)$. Hence there is a minimal 0-certificate or a minimal 1-certificate for f^{i-1} setting x_{n-i+1} to ℓ. Thus by Lemma 1, $g^{i-1}(x_{n-i+1} = \ell) - g^{i-1}(*, \ldots, *) \geq 1$, and hence $g(b^i) - g(b^{i-1}) \geq 1$. Thus the value of g increases by at least 1 for each b^i in the sequence $b^0, b^1, \ldots, b^n$, and so $g(x_n) \geq n$. This proves that $\Gamma(f) \geq n$.

We now need to prove that $\Gamma^1(f) + \Gamma^0(f) \geq n + 1$. Let g' be a 1-goal function for f, and let g'' be a 0-goal value function for f. Considering the sequence $b^0, \ldots, b^n$ again, the above argument, together with Lemma 1, shows that for each $i \in \{1, \ldots, n-1\}$, $g'(b^i) - g'(b^{i-1}) \geq 1$ or $g''(b^i) - g''(b^{i-1}) \geq 1$. Thus $g'(b^{n-1}) + g''(b^{n-1}) \geq n - 1$.

Either the assignment $x_1 = b_1^n$ is in a 1-certificate for f^{n-1}, or in a 0-certificate (or both). Without loss of generality, assume it is in a 1-certificate. Then assignment $x_1 = \neg b_1^n$ is in a 0-certificate for g^{n-1}. Let q' be the goal value of g' and q'' be the goal value for g''. Since b^{n-1} contains neither a 1-certificate nor a 0-certificate for f, $g'(b^{n-1}) \leq q' - 1$ and $g''(b^{n-1}) \leq q'' - 1$. It follows that $n - 1 \leq (q' - 1) + (q'' - 1)$, and so $q' + q'' \geq n + 1$. The follows. □

Furthermore, these bounds can be tight, as illustrated in the following two propositions:

Proposition 1. *The AND, OR, and XOR functions, and their negations, have goal value n.*

Proof. Their goal values must be at least n, by Theorem 1. For AND, consider utility function g whose value equals n on any partial assignment having at least one 0, and whose value equals the number of 1's otherwise. This is a goal function for AND, and there is a dual goal function for OR. For XOR, consider the goal function whose value on a partial assignment is equal to the number of variables set to 0 or 1. □

Proposition 2. *Let $f : \{0,1\}^n \rightarrow \{0,1\}$ be a Boolean k-of-n function. Then $\Gamma^1(f) = k$ and $\Gamma^0(f) = n - k + 1$.*

Proof. For any k-of-n function f, define the 1-goal function g' whose value on any subset of bits is just the minimum of k and the number of bits set to 1. This 1-goal function has goal value k, so $\Gamma^1(f) \leq k$ and by Lemma 1, $\Gamma^1(f) \geq k$. Similarly, define the 0-goal function g'' whose value on any subset of bits is the minimum of $n - k + 1$ and the number of bits set to 0. □

We can further analyze the goal value for k-of-n functions:

Theorem 2. *Let $f : \{0,1\}^n \rightarrow \{0,1\}$ be a Boolean k-of-n function. Then $\Gamma(f) = k(n - k + 1)$.*

Proof. Let g be a goal function for f with goal value Q. Consider a minimal 1-certificate b for f. It has exactly k 1's and no 0's. Since $g(b) = Q$ it follows from the submodularity and monotonicity of g that there must be at least one index i_1 such that $b_{i_1} = 1$ and $g(\{b_{i_1}\}) \geq \frac{1}{k}Q$. Let $t_1 = g(\{b_{i_1}\})$. Thus $g(b) - g(\{b_{i_1}\}) = Q - t_1$. It follows again from the monotonicity and submodularity of g that for some index $i_2 \neq i_1$ where $b_{i_2} = 1$, $g(\{b_{i_1}, b_{i_2}\}) \geq t_1 + \frac{1}{k-1}(Q - t_1)$. Since $t_1 \geq \frac{1}{k}Q$, and $t_1 + \frac{1}{k-1}(Q - t_1) = t_1(1 - \frac{1}{k-1}) + \frac{1}{k-1}Q$, we get that $g(\{b_{i_1}, b_{i_2}\}) \geq \frac{1}{k}(1 - \frac{1}{k-1})Q + \frac{1}{k-1}Q$ so $g(\{b_{i_1}, b_{i_2}\}) \geq Q(\frac{2}{k})$. Settng $t_2 = g(\{b_{i_1}, b_{i_2}\})$, we similarly get i_3 with $g(\{b_{i_1}, b_{i_2}, b_{i_3}\}) \geq \frac{3}{k}Q$ and so forth, until we have $g(\{b_{i_1}, \ldots, b_{i_{k-1}}\}) \geq \frac{k-1}{k}Q$, where $b_{i_1}, \ldots, b_{i_{k-1}}$ are $k - 1$ of the k bits of b that are set to 1. Let b_{i_k} be the remaining bit of b that is set to 1. Let c be the minimal 0-certificate for f such that $c_{i_k} = 0$, and further, $c_i = 0$ for all i where $b_i = *$, and $c_i = *$ for all other i. Let $l = n - k + 1$, which is the number of 0's in c.

Let $t_{k-1} = g(\{b_{i_1}, \ldots, b_{i_{k-1}}\})$, so $t_{k-1} \geq \frac{k-1}{k}Q$. Let d be the assignment such that $d_{i_j} = 1$ for $j \in \{1, \ldots, k-1\}$, and $d_i = 0$ for all other i. Then $g(d) = Q$ and $g(d) - g(\{b_{i_1}, \ldots, b_{i_{k-1}}\}) = Q - t_{k-1}$. Let $Q' = Q - t_{k-1}$.

Let $l = n - k + 1$. It follows from an argument similar to the one above that there are bits $c_{j_1}, \ldots, c_{j_{l-1}}$ equal to 0 such that $g(\{b_{i_1}, \ldots, b_{i_{k-1}}, c_{j_1}, \ldots, c_{j_{l-1}}\}) \geq t_{k-1} + \frac{l-1}{l}Q'$. Let d' be the partial assignment such that $b_{i_1}, \ldots, b_{i_{k-1}} = 1$, $c_{j_1}, \ldots, c_{j_{l-1}} = 0$ and $b_i = *$ for the remaining one variable x_i.

Thus we have $g(d') \geq t_{k-1} + \frac{l-1}{l}Q' = t_{k-1} + \frac{l-1}{l}(Q - t_{k-1})$ and so $g(d') \geq t_{k-1}(1 - \frac{l-1}{l}) + \frac{l-1}{l}Q$. Using the fact that $t_{k-1} \geq \frac{k-1}{k}Q$ we get that $g(d') \geq \frac{k-1}{k}(1 - \frac{l-1}{l})Q + \frac{l-1}{l}Q$ and so $g(d') \geq \frac{kl-1}{kl}Q$. Further, since d' is neither a 0-certificate nor a 1-certificate for f, $g(d') < Q$, and thus $g(d') \leq Q - 1$. It follows that $Q \geq kl = k(n - k + 1)$.

Finally, it follows from previous work (see Lemma 3 in the next section) that there exists a goal function g for f with goal value exactly equal to $kl = k(n - k + 1)$. □

Upper Bound on Goal Value

We do not know the maximum possible goal value of a Boolean function, if goal value is expressed as a function of the number of variables, n. As previously mentioned, Deshpande et al. showed that $f(x_1, \ldots, x_n) = x_1x_2 \vee x_3x_4 \ldots \vee x_{n-1}x_n$ has goal value $2^{.5n}$. A similar argument yields a function with larger goal value.

Theorem 3. *For n a multiple of 3, there is a Boolean function $f(x_1, \ldots, x_n)$ such that $\Gamma(f) \geq 3^{n/3} = 2^{\alpha n}$ where $\alpha = \frac{\log_2 3}{3} \approx .528$.*

Proof. Let n be a multiple of 3, and let $f(x_1, \ldots, x_n) = x_1x_2x_3 \vee x_4x_5x_6 \vee \ldots \vee x_{n-2}x_{n-1}x_n$.

Our proof is by induction on n. The statement is clearly true for $n = 3$, because in this case f has a minimal 1-certificate of size 3, and so its goal value is at least 3.

We now show is is true inductively for larger n. Let g be a goal function for f, with goal value Q. Let b be the all 1's assignment to the variables in V. Clearly $g(\{b_1, b_2, b_3\}) = Q$.

By the monotonicity and submodularity of g, there must be a pair of those three bits for which the value of g is at least $\frac{2}{3}Q$. Assume without loss of generality it is b_1 and b_2 so that $g(\{b_1, b_2\}) \geq \frac{2}{3}Q$. By monotonicity, adding in $x_3 = 0$ cannot decrease utility, so $g(\{b_1, b_2, x_3 = 0\}) \geq \frac{2}{3}Q$ also.

Let $t = g(\{b_1, b_2, x_3 = 0\})$. Consider the induced function f' produced from f by setting $x_1 = 1$, $x_2 = 1$, and $x_3 = 0$. Inductively, this function has goal value at least $3^{(n-3)/3}$.

It follows that $Q - t \geq 3^{(n-3)/3}$, and hence $Q \geq t + 3^{(n-3)/3}$. Since $t \geq \frac{2}{3}Q$, we get that $Q \geq 3^{n/3}$. □

The following is implicit in the work of Deshpande et al.

Lemma 3. *(Deshpande, Hellerstein, and Kletenik 2014) For any Boolean function f, $\Gamma(f) \leq \Gamma^1(f) \cdot \Gamma^0(f)$*

Proof. The proof of the above proposition is constructive, and is based on a standard "OR" construction used previously by (Guillory and Bilmes 2011). Let g_1 be a 1-goal function for f and g_2 be a 0-goal function for f. Let Q_1 and Q_0 be the goal values for these functions. Then function $g(b) = (Q_1 - g_1(b))(Q_2 - g_0(b))$ is a monotone, submodular function where $g(b) = Q_1Q_2$ iff $g_1(b) = Q_1(b)$ or $g_0(b) = Q_0(b)$. The lemma follows. □

As shown in Proposition 1, for the AND function, $\Gamma(f) = n$, and it is easy to show that $\Gamma^0(f) = 1$ while $\Gamma^1(f) = n$. Therefore, for the AND (or the OR) function, we have $\Gamma(f) = \Gamma^1(f) \cdot \Gamma^0(f)$.

In contrast, by Lemma 1 and Proposition 1, when f is the XOR function, $\Gamma(f) = \Gamma^1(f) = \Gamma^0(f) = n$.

We have the following upper bound.

Lemma 4. *Let f be a Boolean function that is not identically 1 or 0. Then for $k \in \{0,1\}$, $\Gamma^k(f) \leq \Gamma(f)$.*

Proof. Let $g : \{0,1,*\}^n \to \mathbb{Z}_{\geq 0}$ be a goal function for f, and let Q be its goal value. Without loss of generality, assume $k = 1$. Let $g^1 : \{0,1,*\}^n \to \mathbb{Z}_{\geq 0}$ be such that for $b \in \{0,1,*\}^n$, $g^1(b) = Q - 1$ if b contains a 0-certificate of f, and $g^1(b) = g(b)$ otherwise. Clearly $g^1(b) = Q$ iff b contains a 1-certificate of f. Using the monotonicity and submodularity of g, and the fact that any extension of a 0-certificate is also a 0-certificate, it is straightforward to show that g^1 is monotone and submodular. □

Goal Value of Read-Once Formulas

In this section, we discuss bounds on the goal value of functions expressed by read-once formulas.

Given a function $g : \{0,1,*\}^n \to \mathbb{Z}_{\geq 0}$, we say that g *gives no value to 0's* if for all $b \in \{0,1,*\}$, $g(b) = g(b')$, where b' is the partial assignment produced from b by changing all 0's in b to $*$'s. We say that g *gives no value to 1's* if the analogous property holds.

The following lemma will be useful.

Lemma 5. *Let f be a monotone Boolean function. Then there is a 1-goal function g for f that gives no value to 0's, whose goal value is $\Gamma^1(g)$. Similarly, there is a 0-goal function g for f that gives no value to 1's, whose goal value is $\Gamma^0(g)$.*

Proof. We prove this for the 1-goal function; the proof is symmetric for 0-goal function. Let g be a 1-goal utility function for f with 1-goal value $\Gamma^1(f)$. For $b \in \{0,1,*\}$, let b' be the partial assignment produced from b by changing all 0's in b to $*$'s. Define a related utility function h such that $h(b) = g(b')$. It is straightforward to verify that because f is monotone, h is also a 1-goal utility function for f, with the same 1-goal value as g, namely $\Gamma^1(f)$. □

In what follows, if ϕ is a DNF formula, we use $size(\phi)$ to denote the number of terms of ϕ. (If ϕ is the constant formula 0 or 1, $size(\phi) = 0$.)

Lemma 6. *Let F be a read-once formula on variable set $V = \{1, \ldots, n\}$, and let f be the monotone Boolean function that it computes. Let ϕ be the canonical DNF formula for f. Let $h : 2^V \to \mathbb{Z}_{\geq 0}$ be a monotone, submodular set function, such that for some $Q > 0$, $h(S) = Q$ if S contains a maxterm of f, and $h(S) < Q$ otherwise.*

Then the following 2 properties hold:

i) For any variable $x_i \in V$, $h(\{x_i\}) \geq$ number of terms of ϕ containing x_i

ii) $h(V) \geq size(\phi)$.

Proof. We do induction on n. In the base case, $n = 1$. Formula F then consists of a single node labeled x_1 and $\phi = x_1$. Since g is a 0-goal utility function, $h(\emptyset) < h(\{x_1\})$, and therefore $h(V) = h(\{x_1\}) \geq 1 =$ number of terms of ϕ containing x_1 = $size(\phi)$.

For the induction step, let $k \geq 0$ and assume that the claim is true for $n = k$. Suppose F has size $n = k + 1$.

We first prove (i). Let x_i be an arbitrary variable of F.

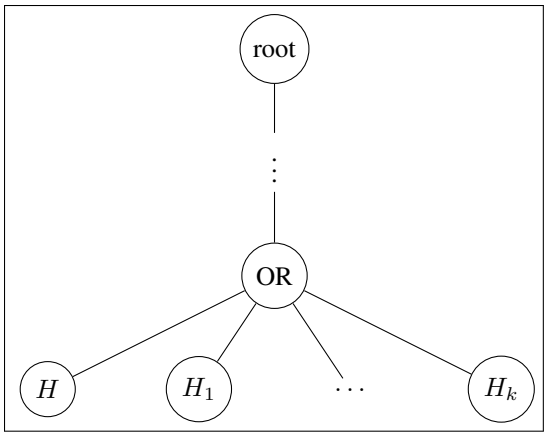

Figure 1: x_i is in one of the H_j

There are two cases.

Case 1: There is an ancestor of x_i in F that is labeled OR

Let t be an ancestor of x_i in F that is labeled OR. Let H be a subtree of F that is rooted at a child of t, such that H does not contain x_i (that is, the path from t down to x_i does not go through H). See Figure 1.

Let S denote the variables in H. Because t is labeled OR, and the terms of ϕ correspond to the minterms of f, if a term of ϕ contains any variables from S, it cannot contain x_i.

Let $V' = V - S$. Let F' denote the induced read-once formula produced from F by setting the variables in H to 0. Setting the variables in H removes the subtree H from F. In order to produce a read-once formula in valid form, we follow the following procedure: if after removing H, t has only one child, t is deleted. If t had a parent, the remaining child of t will become the child of the previous parent of t; if t did not have a parent, the remaining child becomes the new root.

Let ϕ' be the canonical DNF formula for F'. Formula ϕ' is the formula produced from ϕ by deleting all terms containing at least one variable in H. Since x_i does not appear in any of the deleted terms, the number of terms of ϕ that contain x_i is equal to the number of terms of ϕ' that contain x_i.

We define a set function $h' : 2^{V'} \to \mathbb{Z}_{\geq 0}$ where for all subsets $R \subseteq V'$, $h'(R) = h(R \cup S) - h(S)$. Function h' is clearly monotone and submodular. Since F is read-once and the parent of subtree H is labeled OR, set $R \subseteq V'$ contains a maxterm of F' iff $R \cup H$ contains a maxterm of F. It follows that $h'(R) = h'(V') = h(V) - h(S)$ if R is a maxterm of h', and $h'(R) < h(V) - h(S)$ otherwise.

By the inductive hypothesis, $h'(\{x_i\}) \geq$ (the number of terms of ϕ' containing x_i). Since the number of terms of ϕ' containing x_i is equal to the number of terms of ϕ containing x_i, $h'(\{x_i\}) = h(\{x_i\} \bigcup S) - h(S) \geq$ (number of terms of ϕ containing x_i). By the submodularity of h, $h(\{x_i\}) \geq h(\{x_i\} \cup S) - h(S)$, so $h(\{x_i\}) \geq$ (number of terms of ϕ containing x_i).

Case 2: All ancestors of x_i in F are labeled AND.

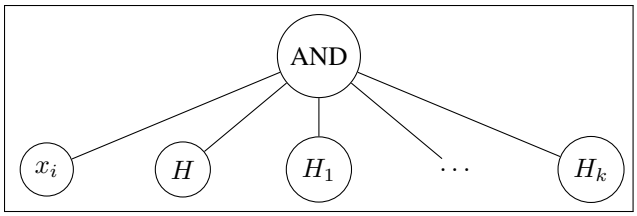

Figure 2: merging of the ANDs

Because all ancestors of x_i are AND, we can merge the ANDs, and without loss of generality, we can assume that x_i is a child of the root of F, which is labeled AND. See illustration in Figure 2.

Let H be a subtree of F rooted at a sibling of x_i, and let S denote the variables in H. Let $V' = V - \{x_i\}$.

Clearly $\{x_i\}$ is a maxterm of F, and every maxterm of H is a maxterm of F. Consider the read-once formula induced from F by setting x_i to 1. (Note: if x_i and H were the only children of the root, then the resulting read-once formula is just H.) Call it F'. Since the root is AND, the canonical DNF for f contains x_i in every term. Let ϕ' be the canonical DNF for F'. It follows that ϕ' is produced from F by deleting x_i from each term of ϕ (and no redundant or subsumed terms are created), and that $size(\phi) = size(\phi')$.

Consider the restriction of utility function h to the variables in V'. Call the restriction h'. That is, $h' : 2^{V'} \to \mathbb{Z}_{\geq 0}$ where for all $R \subseteq V'$, $h'(R) = h(R)$. Because h is monotone and submodular, so is h'. Clearly any $R \subseteq V'$ is a maxterm of F' iff it is a maxterm of F. Also, since S is a subset of V', and S contains maxterms of H and hence of F and F', $h'(V') = h(V)$. For $R \subseteq V'$, $h'(R) = h'(V') = h(V)$ if R is a maxterm of F', else $h'(R) < h(V)$. Since F' has fewer variables than F, by the inductive hypothesis, $h'(V') \geq size(\phi') = size(\phi)$. Finally, since $\{x_i\}$ is a maxterm of F, and h is a 0-utility function for f, $h(V) = h(\{x_i\})$. We thus have $h(\{x_i\}) = size(\phi) =$ number of terms containing x_i, since every term of ϕ contains x_i.

This completes the proof of (i) for Case 2.

We now prove Property (ii) of the claim, in the general case.

If every variable x_i of F is such that $\{x_i\}$ is a maxterm of F, then F is just the AND of all the variables, and (ii) follows immediately from (i).

Suppose there is a variable x_i of F where $\{x_i\}$ is not a maxterm of F.

By (i), $h(\{x_i\}) \geq$ number of terms of ϕ containing x_i. Let F' be the read-once formula produced from F by setting x_i to 0. Since $\{x_i\}$ is not a maxterm of F, F' has at least one variable. Let ϕ' be the canonical DNF for F'. Let $V' = V - \{x_i\}$. Let $h' : 2^{V'} \to \mathbb{Z}_{\geq 0}$ where for $R \subseteq V'$, $h'(R) = h(R \cup \{x_i\}) - h(\{x_i\})$. DNF formula ϕ' is the formula you get by deleting all terms in ϕ that contain x_i. It is easy to see that h' satisfies the conditions of the claim, for formula F'. Therefore, by induction, $h'(V') \geq$ the number of terms of ϕ'. Therefore, we have that $h'(V') = h(V) - h(\{x_i\})$, so $h(V) = h'(V') + h(\{x_i\}) \geq$ (number of terms of ϕ not containing x_i) + (number of terms of ϕ containing x_i) = $size(\phi)$. □

Theorem 4. *If f is a Boolean function that is computed by a read-once formula, then $\Gamma^0(f) \geq \mathrm{ds}(f)$, $\Gamma^1(f) \geq cs(f)$, and $\Gamma(f) \geq \max\{ds(f), cs(f)\}$.*

Proof. Let ϕ be the canonical DNF formula for f. Let $size(\phi)$ denote the number of terms in ϕ.

We prove that $\Gamma^0(f) \geq ds(f)$. A dual argument shows that $\Gamma^1(f) \geq \mathrm{cs}(f)$. It then follows from Lemma 4 that $\Gamma(f) \geq \max\{\mathrm{ds}(f), \mathrm{cs}(f)\}$.

Let g be a 0-goal function for f, and let Q be its 0-goal value. By Lemma 5, we may assume that g gives no value to 1's. We can therefore associate with g a utility function h defined on the subsets of V, such that for $S \subseteq V$, $h(S) = g(b)$, where b is the partial assignment setting the variables in S to 1, and all other variables to $*$. Clearly, $h(S) = Q$ iff set S contains a maxterm of h (that is, there is a subset $S' \subseteq S$ such that S' is a maxterm of f).

By Lemma 6, $h(V) \geq \mathrm{ds}(f)$. Therefore, $Q \geq \mathrm{ds}(f)$ and hence $\Gamma^0(f) \geq \mathrm{ds}(f)$.

□

For a function f represented by a read-once formula, we thus have $\max\{\mathrm{ds}(f), \mathrm{cs}(f)\} \leq \Gamma(f) \leq \mathrm{ds}(f) \cdot \mathrm{cs}(f)$. There is at most a quadratic gap between these upper and lower bounds.

Goal Value and Other Measures of Boolean Function Complexity

In this section, we relate the goal value of a monotone Boolean function to decision tree rank and polynomial threshold degree.

A decision tree computes a monotone submodular function h as follows: It has internal nodes labeled with variables $x_i \in V$. Each leaf of the tree is labeled with an element of $\{0, 1 \ldots, d\}$. In using the tree to compute the value of h on input $S \subseteq V$, for each internal node labeled x_i, you branch right if x_i is in S, and left if not.

Lemma 7. *If $h : 2^V \to \{0, 1, \ldots, d\}$ is a monotone, submodular set function, then there is a decision tree computing h that has rank d.*

Proof. It was shown in (Feldman, Kothari, and Vondrák 2013) that for submodular h (not necessarily monotone), there is a decision tree computing h of rank at most $2d$. Here we give a better bound on rank when monotonicity holds.

The proof is by induction on n and d. It is clearly true if $n = 0$ and $d = 0$, and in particular, it is true for $n + d = 0$. Let $s > 0$, and assume true for $n + d < s$. We now show true for $n + d = s$.

If h is identically equal to a single value in $\{0, 1, \ldots, d\}$, then the decision tree can consist of a single node labeled with that value.

Otherwise, there exists a subset $S \subseteq V$ such that $h(S) \neq h(\emptyset)$. By monotonicity, $h(S) > h(\emptyset)$. Let S be such that $h(S)$ is maximized. If $h(S) < d$, then a decision tree of rank at most d must exist by induction. Suppose $h(S) = d$.

By submodularity and monotonicity, there exits $x_i \in S$ such that $h(\{x_i\}) > h(\emptyset)$, so $h(\{x_i\}) \geq 1$. We construct a decision tree for h by first putting x_i in the root. The left subtree of this tree needs to compute the function $h' : 2^{V-\{x_i\}} \rightarrow \{0, 1, \ldots, d\}$ such that $h'(S) = h(S)$ for all $S \subseteq V - \{x_i\}$. This is a function of subsets of $n-1$ variables, and function h' is monotone and submodular, and thus by induction, can be computed by a tree of rank at most d.

The right subtree needs to compute the function $h'' : 2^{V-\{x_i\}} \rightarrow \{0, 1, \ldots, d\}$ such that $h''(S) = h(S \cup \{x_i\})$ for all $S \subseteq V$.

We now show that there is a decision tree computing h'' that has rank at most $d-1$. Consider the function $\hat{h}$ defined on $2^{V-\{x_i\}}$ such that for $S \subseteq V$, $\hat{h}(S) = h(S \cup \{x_i\}) - h(\{x_i\})$ for all x_i in S. Since h is monotone, so is $\hat{h}$. Further, since $h(\{x_i\}) \geq 1$ and $h(S \cup \{x_i\}) \leq d$, it follows that $\hat{h}$ can be viewed as mapping to $\{0, 1, \ldots, d-1\}$. Function $\hat{h}$ is defined on $n-1$ variables, and since it is monotone and submodular, by induction, it can be computed by a tree of rank at most $d-1$. If modify the tree by adding the value $h(\{x_i\})$ to the value in each leaf of the tree, the resulting tree will compute h''.

We have thus shown that we can construct a tree computing h such that the two subtrees of the root have rank d and $d-1$ respectively. Such a tree has rank d. □

Theorem 5. *Let f be a monotone Boolean function, and let $d = \min\{\Gamma^1(f), \Gamma^0(f)\}$. Then there is a decision tree of rank at most d computing f. There is a polynomial-threshold function of degree at most d computing f.*

Proof. Consider a 1-goal function g for a monotone Boolean function f, whose goal value is d. By Lemma 5, we can assume that the value of g only increases on bits that are set to 1, and stays the same on bits that are set to 0. Let $V = \{x_1, \ldots, x_n\}$ Define a set function $h : 2^V \rightarrow \{0, \ldots, d\}$, such that for all $S \subseteq V$, $h(S)$ is equal to the value of g on the partial assignment setting the variables in S to 1, and all other variables to 0.

By Lemma 7, there is a decision tree computing h with rank at most d. This same tree must compute the function g (if at an internal node labeled x_i, you branch left if $x_i = 0$, and right if $x_i = 1$). For each leaf of the tree, change the value of the leaf to 0 if it is labeled with a value in $0, \ldots, d-1$, and to 1 if it is labeled with the value d. Since d is the 1-goal value for g, on any assignment in 2^V, the value computed by the tree will be 1 iff the assignment contains a 1-certificate for f. It follows that a tree of rank at most d computes f.

By (Blum 1992), a function that can be computed by a tree of rank d has an equivalent d-decision list. By a similar argument to that used in (Ehrenfeucht et al. 1989), any function expressible by a d-decison list is also expressible as a degree-d polynomial threshold function.

□

It follows from the above and from Lemma 4 that the goal value of a monotone Boolean function is lower bounded by its polynomial threshold degree.

Open Questions

There remain many open questions concerning the goal value of Boolean functions. We list some here:

- As a function of n, what is the largest possible goal value for a Boolean function?
- What is the expected goal value of a random Boolean function?
- Are there any techniques for deriving good lower bounds for $\Gamma(f)$?
- Is our lower bound on $\Gamma(F)$ where F is a read-once formula, actually tight?
- Is goal value polynomially related to some known measure of the complexity of a Boolean function?

References

Allen, S.; Hellerstein, L.; Kletenik, D.; and Ünlüyurt, T. 2014. Evaluation of DNF formulas. *CoRR* abs/1310.3673v3.

Blum, A. 1992. Rank-r decision trees are a subclass of r-decision lists. *Information Processing Letters* 42(4):183–185.

Buhrman, H., and Wolf, R. D. 1999. Complexity measures and decision tree complexity: A survey. *Theoretical Computer Science* 288:2002.

Deshpande, A., and Hellerstein, L. 2008. Flow algorithms for parallel query optimization. In *Proceedings of the 24th International Conference on Data Engineering (ICDE)*, 754–763.

Deshpande, A.; Hellerstein, L.; and Kletenik, D. 2013. Approximation algorithms for stochastic boolean function evaluation and stochastic submodular set cover. *CoRR* abs/1303.0726v2.

Deshpande, A.; Hellerstein, L.; and Kletenik, D. 2014. Approximation algorithms for stochastic boolean function evaluation and stochastic submodular set cover. In *SODA 2014*, 1453–1466. SIAM.

Ehrenfeucht, A., and Haussler, D. 1989. Learning decision trees from random examples. *Information and Computation* 82(3):231 – 246.

Ehrenfeucht, A.; Haussler, D.; Kearns, M.; and Valiant, L. 1989. A general lower bound on the number of examples needed for learning. *Information and Computation* 82(3):247–261.

Feldman, V.; Kothari, P.; and Vondrák, J. 2013. Representation, approximation and learning of submodular functions using low-rank decision trees. In *Conference on Learning Theory*, 711–740.

Golovin, D., and Krause, A. 2011. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research* 42:427–486.

Guillory, A., and Bilmes, J. 2011. Active semi-supervised learning using submodular functions. In *UAI*, 274–282.

Ibaraki, T., and Kameda, T. 1984. On the optimal nesting order for computing n-relational joins. *ACM Trans. Database Syst.*

Kaplan, H.; Kushilevitz, E.; and Mansour, Y. 2005. Learning with attribute costs. In *Symposium on the Theory of Computing*, 356–365.

Krishnamurthy, R.; Boral, H.; and Zaniolo, C. 1986. Optimization of nonrecursive queries. In *Twelfth International Conference on Very Large Data Bases (VLDB)*, 128–137.

Rivest, R. L. 1987. Learning decision lists. *Machine learning* 2(3):229–246.

Srivastava, U.; Munagala, K.; Widom, J.; and Motwani, R. 2006. Query optimization over web services. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB)*, 355–366.

Ünlüyurt, T. 2004. Sequential testing of complex systems: a review. *Discrete Applied Mathematics* 142(1-3):189–205.