Electronic Colloquium on Computational Complexity, Report No. 11 (1999)

APPROXIMATIONS BY OBDDs AND THE VARIABLE ORDERING PROBLEM

Matthias Krause[1], Petr Savický[2] and Ingo Wegener[3]

[1] Theoretische Informatik, Univ. Mannheim, 68131 Mannheim, Germany
e-mail: krause@informatik.uni-mannheim.de
[2] Institute of Computer Science, Academy of Sciences of Czech Republic
Pod vodárenskou věží 2, 182 07 Praha 8, Czech Republic
e-mail: savicky@uivt.cas.cz
[3] FB Informatik, LS 2, Univ. Dortmund, 44221 Dortmund, Germany
e-mail: wegener@ls2.cs.uni-dortmund.de

Abstract. Ordered binary decision diagrams (OBDDs) and their variants are motivated by the need to represent Boolean functions in applications. Research concerning these applications leads also to problems and results interesting from theoretical point of view. In this paper, methods from communication complexity and information theory are combined to prove that the direct storage access function and the inner product function have the following property. They have linear π-OBDD size for some variable ordering π and, for most variable orderings π', all functions which approximate them on considerably more than half of the inputs, need exponential π'-OBDD size. These results have implications for the use of OBDDs in experiments with genetic programming.

1 INTRODUCTION

Branching programs (BPs) or binary decision diagrams (BDDs), which is just another name, are representations of Boolean functions $f \in B_n$, i.e., $f\colon \{0,1\}^n \to \{0,1\}$. They are compact but not useful for manipulations of Boolean functions, since operations like satisfiability test, equivalence test or minimization lead to hard problems. Bryant [6] has introduced π-OBDDs (ordered BDDs), since they can be manipulated efficiently (see [7] and [19] for surveys on the areas of application).

Definition 1. A permutation π on $\{1,\dots,n\}$ describes the variable ordering $x_{\pi(1)},\dots,x_{\pi(n)}$. A π-OBDD is a directed acyclic graph $G=(V,E)$ with one source. Each sink is labelled by a Boolean constant and each inner node by a Boolean variable. Inner nodes have two outgoing edges one labelled by 0 and the other by 1. If an edge leads from an x_i-node to an x_j-node, then $\pi^{-1}(i)$ has to be smaller than $\pi^{-1}(j)$, i.e., the edges have to respect the variable ordering. The π-OBDD represents a Boolean function $f \in B_n$ defined in the following way. The input a activates, for x_i-nodes, the outgoing a_i-edge. Then $f(a)$ is equal to the label of the sink reached by the unique activated path starting at the source. The size of G is measured by the number of its nodes. An OBDD is a π-OBDD for an arbitrary π.

One-way communication complexity (see e.g. [11], [14]) leads to lower bounds for OBDDs. This method is almost the same as counting the number of subfunctions of

[1] Supported by DFG grant Kr 1521/3-1.
[2] The research was partially supported by GA of the Czech Republic, Grant No. 201/98/0717.
[3] Supported by DFG grant We 1066/8-1 and by the DFG as part of the Collaborative Research Center "Computational Intelligence" (531).

ISSN 1433-8092

f if the first variables according to the variable ordering are replaced by constants. There are functions, for which the OBDD size is very sensitive to the chosen variable ordering. Moreover, given a π-OBDD for a function f, it is NP-hard to find an optimal variable ordering for f, see [5], or even to approximate the optimal variable ordering, see [17].

We will need the following two functions.

Definition 2. i) For $n = 2^k$, the direct storage access function (or multiplexer) on $k + n$ variables is the function $\mathrm{DSA}_n(a_0, \dots, a_{k-1}, x_0, \dots, x_{n-1}) = x_{||a||}$, where $||a||$ is the number whose binary representation is $(a_0, \dots, a_{k-1})$.
ii) For any even n, the inner product function on n variables is the function $\mathrm{IP}_n(x_1, \dots, x_n) = x_1x_2 \oplus x_3x_4 \oplus \dots \oplus x_{n-1}x_n$.

Clearly, these two functions have π-OBDD of size $O(n)$ for the ordering of the varibles used in the definition of the functions. On the other hand, they need exponential π-OBDD size for most of the variable orderings (a fraction of $1 - n^{-\varepsilon}$ for DSA_n and even a fraction of $1 - 2^{-\varepsilon n}$ for IP_n, see [20]). Another example of a function with a similar property, so-called disjoint quadratic form, may be obtained by replacing $\oplus$ by disjunction in the expression describing IP_n.

These results are stated for error-free representations of f and, up to now, to the best of our knowledge, nobody has looked at representations of approximations of f from a theoretical point of view. In this paper, we investigate the influence of the variable ordering for approximate representations of functions. If not stated otherwise, by a random input $\tilde{x}$ we mean an input $\tilde{x}$ chosen from the uniform distribution on $\{0, 1\}^n$.

Definition 3. *A function $g \in B_n$ is a c-approximation of $f \in B_n$ if $Pr(f(\tilde{x}) = g(\tilde{x})) \geq c$ for a random input $\tilde{x}$ chosen from the uniform distribution.*

One of the two constant functions 0 and 1 always is a 1/2-approximation. Hence, we consider c-approximations for $c > 1/2$.

We prove the following strengthenings of the previously mentioned lower bounds on the π-OBDD complexity of DSA_n and IP_n for a random ordering. For most of the orderings π, every function that is a $(1/2 + \varepsilon)$-approximation of DSA_n or IP_n, where ε, $0 < \varepsilon < 1/2$ is any constant, requires a π-OBDD of exponential size. For the exact formulation of the result for DSA_n see Theorem 1. The exact result for IP_n is remarkably stronger then the formulation above, since it is proved for better parameters. For the exact formulation see Theorem 2. On the other hand, the result for DSA_n is of particular interest for genetic programming, since, recently, DSA_n is frequently used in experiments. The proof combines methods from one-way communication complexity and information theory.

The problem of approximation is motivated by experiments in genetic programming using OBDDs, where one searches for a good approximation of an unknown function given by examples. Our results have consequences for the situation that the unknown function has a small OBDD for some ordering, but this ordering is not known. For more details see Section 4.

For completeness, we present also an example of a function that is hard to approximate for any ordering.

2 THE DIRECT STORAGE ACCESS FUNCTION

First, we state the result informally. There are only a few variable orderings π which allow an approximation g_n of DSA_n which is essentially better than the trivial approximations by the constants 0 and 1 (which are 1/2-approximations) and which, moreover, has a π-OBDD size growing not exponential.

Theorem 1. *Let $0 < \delta < \varepsilon$. For every large enough n, the following property holds for a fraction of at least $1 - n^{-2\varepsilon^2/\ln 2}$ of the variable orderings π for DSA_n. Each function which is a $(\frac{1}{2} + \varepsilon + n^{-(\varepsilon-\delta)/2})$-approximation of DSA_n has a π-OBDD size which is bounded below by e^{n^δ}.*

The proof of this theorem is splitted into Lemmas 1 and 2. Recall that an ordering π for DSA_n is a permutation of $n + k$ variables, where $k = \log n$. In order to simplify the terminology, we assume that the first $n' =_{\mathrm{def}} \lfloor (1 - 2\varepsilon)n \rfloor$ variables according to π are given to Alice and the other ones to Bob. First, we derive a property of random variable orderings π.

Lemma 1. *With probability at least $1 - n^{-2\varepsilon^2/\ln 2}$, Alice obtains at most $(1 - \varepsilon)k$ address variables, i.e., a-variables.*

Proof. The random variable ordering can be produced as follows. We take the k address variables and randomly choose for them one after another a free position among the $n + k$ possible positions. Then we continue in the same way with the n data variables, i.e., the x-variables. During the first k steps of this process, there are always at most $(1 - 2\varepsilon)n$ free positions among the first $n' = \lfloor (1 - 2\varepsilon)n \rfloor$ positions and at least n open positions at all. Hence, the probability of each address variable to be given to Alice, is at most $1 - 2\varepsilon$. We can upper bound the probability that Alice gets more than $(1-\varepsilon)k$ address variables by the probability of at least $(1-\varepsilon)k$ successes in k independent Bernoulli trials with success probability $1 - 2\varepsilon$.

The expected number of successes $\mathrm{E}[Z]$ equals $(1 - 2\varepsilon)k$. By Chernoff's bound, we obtain

$$\Pr(Z \geq (1 - \varepsilon)k) = \Pr(Z \geq \mathrm{E}[Z] + \varepsilon k) \leq e^{-2\varepsilon^2 k} = n^{-2\varepsilon^2/\ln 2}.$$

□

In the following, we fix a variable ordering π where Alice gets at most $(1 - \varepsilon)k$ address variables. She also gets at least $(1-2\varepsilon)n-k$ data variables. If Alice's address variables are fixed, there are at least n^ε data variables left which may describe the output. On the average, at least $(1 - 2\varepsilon)n^\varepsilon - o(1)$ of these variables are given to Alice. In order to enable Bob to compute the output exactly, Alice has to send him the value of her address variables and those data variables which can describe the output. If the information given from Alice is much smaller than this, Bob can compute the value of DSA_n only with probability close to $1/2$. The information given from Alice to Bob is measured by the logarithm of the size of a π-OBDD computing the function DSA_n.

For a rigorous argument, let π be an ordering and let A (resp. B) be the set of address variables given in π to Alice (resp. Bob) and let X (resp. Y) be the set of data variables given in π to Alice (resp. Bob). Clearly, $|A \cup X| = n'$ and every computation in any π-OBDD reads first (some of) the variables in $A \cup X$ and then (some of) the variables in $B \cup Y$. Let g be a function represented by a π-OBDD G of size s. Because of the definition of c-approximations, we consider random inputs $(\tilde{a}, \tilde{b}, \tilde{x}, \tilde{y})$ where $\tilde{a}$ is a random setting of variables in A, etc. In this situation, the following holds.

Lemma 2. $\Pr(\mathrm{DSA}_n(\tilde{a}, \tilde{b}, \tilde{x}, \tilde{y}) = g(\tilde{a}, \tilde{b}, \tilde{x}, \tilde{y})) \leq 1 - \frac{|X|}{2n} + \frac{1}{2n}\left(2 \cdot 2^{|A|}|X| \ln s\right)^{1/2}$.

Before proving Lemma 2, let us demonstrate its application by proving Theorem 1.

Proof of Theorem 1. Recall that $k = \log n$ and let $s < e^{n^\delta}$. For every ordering π, we have $(1-2\varepsilon)n-k-1 \leq |X| \leq n$. Moreover, Lemma 1 implies that with probability at least $1 - n^{-2\varepsilon^2}$, we have $|A| \leq (1-\varepsilon)k$. By substituting these estimates into the bound from Lemma 2, we obtain that the probability that DSA_n and g have the same value is at most $\frac{1}{2} + \varepsilon + \frac{1}{\sqrt{2}} n^{-(\varepsilon-\delta)/2} + \frac{\log n}{2n} < \frac{1}{2} + \varepsilon + n^{-(\varepsilon-\delta)/2}$. This implies the theorem. $\square$

For proving Lemma 2, we need some more notation. Let $H(U)$ be the entropy of a random variable U and $H(U \mid E)$ resp. $H(U \mid V)$ the entropy of U given an event E or a random variable V resp. Moreover, let $H^*(x) = -x \log x - (1-x)\log(1-x)$ for $x \in (0,1)$.

For each (a, b, y), let

$$q(a,b,y) = \Pr(\mathrm{DSA}_n(a,b,\tilde{x},y) = g(a,b,\tilde{x},y))$$

for random assignments $\tilde{x}$ to the variables in X. The probability, we are interested in, is the average of all $q(a,b,y)$. Let $q(a,b)$ denote the average of $q(a,b,y)$ over all possible y and, similarly, let $q(a)$ denote the average of $q(a,b,y)$ over all possible b and y. Moreover, for each partial input a, let I_a be the set of partial inputs b such that the variable $x_{\|(a,b)\|}$ or $x_{a,b}$, for simplicity, is given to Alice. Note that $H^*(x)$ is maximal for $x = 1/2$ and the maximum is equal to 1. Hence, if $|I_a| \gg \log s$, the next lemma implies that for most of $b \in I_a$, $q(a,b)$ is close to $1/2$.

Lemma 3. *For every a, we have*

$$\sum_{b \in I_a} H^*(q(a,b)) \geq |I_a| - \log s.$$

Proof. Consider the π-OBDD computing the function g described before Lemma 2. For any settings a, x, let $h(a,x)$ be the first node, where the computation for a, x reaches a node testing a variable in $B \cup Y$ or a sink. Note that a computation for a, b, x, y depends on a, x only via $h(a,x)$. This means, there is a function $\Phi_{b,y}$ such that $g(a,b,x,y) = \Phi_{b,y}(h(a,x))$. Note that the size of the range of h is at most s.

Besides well-known information theoretical inequalities we use the following one whose proof is postponed to the end of the section.

Claim. Let U and V be random variables taking values in $\{0,1\}$. Then $H^*(\Pr(U = V)) \geq H(U \mid V)$.

If (a,b,y) is fixed and $b \in I_a$, DSA_n outputs $x_{a,b}$. Using the claim and the fact that $H(U \mid f(V)) \geq H(U \mid V)$ for each function f, we conclude

$$\begin{aligned} H^*(q(a,b,y)) &= H^*(\Pr(\tilde{x}_{a,b} = \Phi_{b,y}(h(a,\tilde{x}))) \\ &\geq H(\tilde{x}_{a,b} \mid \Phi_{b,y}(h(a,\tilde{x}))) \geq H(\tilde{x}_{a,b} \mid h(a,\tilde{x})). \end{aligned}$$

Now we use the fact $H(U_1 \mid V) + \ldots + H(U_r \mid V) \geq H((U_1, \ldots, U_r) \mid V)$ for $\tilde{x}_{a,b}$, $b \in I_a$, and the vector $\tilde{x}_a$ of these random variables. This implies

$$\sum_{b \in I_a} H(\tilde{x}_{a,b} \mid h(a,\tilde{x})) \geq H(\tilde{x}_a | h(a,\tilde{x})).$$

In the next step we apply the equalities $H(U \mid V) = H(U,V) - H(V)$ and $H(U, f(U)) = H(U)$ to obtain

$$H(\tilde{x}_a \mid h(a,\tilde{x})) = H(\tilde{x}_a, h(a,\tilde{x})) - H(h(a,\tilde{x})) = H(\tilde{x}_a) - H(h(a,\tilde{x})).$$

We have $H(h(a,\tilde{x})) \leq \log s$, since there are only s different possibilities for $h(a,\tilde{x})$. The random variables $\tilde{x}_{a,b}$, $b \in I_a$, are independent and take values in $\{0,1\}$, i.e., $\tilde{x}_a$ is uniformly distributed over $\{0,1\}^{|I_a|}$ and $H(\tilde{x}_a) = |I_a|$. This implies

$$H(\tilde{x}_a \mid h(a,\tilde{x})) \geq |I_a| - \log s.$$

Putting all our considerations together, we obtain

$$\sum_{b\in I_a} H^*(q(a,b,y)) \geq |I_a| - \log s.$$

The function H^* is concave. Hence, this inequality implies Lemma 3. $\Box$

Proof of Lemma 2. Let $\Delta(a,b) = q(a,b) - \frac{1}{2}$. Then we apply the inequality $H^*(\frac{1}{2}+t) \leq 1-(2/\ln 2)t^2$ (estimate Taylor's expansion using the second derivative) to obtain

$$\sum_{b\in I_a} H^*(q(a,b)) = \sum_{b\in I_a} H^*\left(\frac{1}{2}+\Delta(a,b)\right) \leq |I_a| - (2/\ln 2)\sum_{b\in I_a}\Delta(a,b)^2.$$

Together with Lemma 3, we get

$$\frac{1}{2}\ln s \geq \sum_{b\in I_a}\Delta(a,b)^2.$$

Using Cauchy's inequality, we obtain

$$\sum_{b\in I_a}|\Delta(a,b)| \leq \left(|I_a|\sum_{b\in I_a}\Delta(a,b)^2\right)^{1/2} \leq \left(\frac{1}{2}|I_a|\ln s\right)^{1/2}.$$

Recall that $q(a)$ is the average of all $q(a,b)$. Since b may take $2^{|B|}$ values, we get

$$\begin{aligned} q(a) &= \frac{1}{2^{|B|}}\left(\sum_{b\notin I_a} q(a,b) + \sum_{b\in I_a} q(a,b)\right) \\ &\leq \frac{1}{2^{|B|}}\left(2^{|B|} - \frac{1}{2}|I_a| + \sum_{b\in I_a}\Delta(a,b)\right) \\ &\leq 1 - 2^{-|B|-1}(|I_a| - (2|I_a|\ln s)^{1/2}) = \psi(|I_a|), \end{aligned}$$

where $\psi(t) =_{\text{def}} 1 - 2^{-|B|-1}(t - (2t\ln s)^{1/2})$. The function ψ is concave. Let $a_1,\ldots,a_m$, $m = 2^{|A|}$, be the possible values of a. Then

$$\frac{1}{m}\sum_{1\leq i\leq m} q(a_i) \leq \frac{1}{m}\sum_{1\leq i\leq m}\psi(|I_{a_i}|) \leq \psi\left(\frac{1}{m}\sum_{1\leq i\leq m}|I_{a_i}|\right) = \psi(|X|/2^{|A|}).$$

The last equality follows, since, by definition, the sum of all $|I_{a_i}|$ equals $|X|$. The left-hand side of the above inequality is the average of all $q(a)$ and this is the average of all $\Pr(\text{DSA}(a,b,\tilde{x},y) = g(a,b,\tilde{x},y))$ and, therefore, equal to $\Pr(\text{DSA}(\tilde{a},\tilde{b},\tilde{x},\tilde{y}) = g(\tilde{a},\tilde{b},\tilde{x},\tilde{y}))$. We have proved that this probability is bounded above by

$$\psi(|X|/2^{|A|}) = 1 - \frac{|X|}{2\cdot 2^{|A|+|B|}} + \frac{1}{2\cdot 2^{|A|+|B|}}(2\cdot 2^{|A|}|X|\ln s)^{1/2}.$$

Since A and B are a partition of the $\log n$ address variables, we have $2^{|A|+|B|} = n$ and Lemma 2 follows. $\Box$

Proof of the claim. Since U and V take values in $\{0,1\}$,

$$\Pr(U=V) = \sum_{\alpha\in\{0,1\}}\Pr(U=\alpha\,|\,V=\alpha)\Pr(V=\alpha).$$

The concavity of H^* implies

$$H^*(\Pr(U=V)) \geq \sum_{\alpha\in\{0,1\}} H^*(\Pr(U=\alpha \mid V=\alpha))\Pr(V=\alpha).$$

Since $H^*(x) = H^*(1-x)$, we obtain

$$H^*(\Pr(U=0 \mid V=\alpha)) = H^*(\Pr(U=1 \mid V=\alpha)) = H(U \mid V=\alpha)$$

and

$$H^*(\Pr(U=V)) \geq \sum_{\alpha\in\{0,1\}} H(U \mid V=\alpha)\Pr(V=\alpha) = H(U \mid V).$$

□

Let us add some comments to this result. A random variable ordering for DSA_n is with a probability of at least $n^{-\log n}$ optimal, i.e., all address variables are tested before each data variable. If we consider cuts as in our proof, Alice gets all address variables with a probability which is approximately $n^{\log(1-2\varepsilon)}$. Therefore, we need another approach to improve the result with respect to the fraction of variable orderings but one cannot obtain a result for an exponentially small fraction. Bob gets at least $2\varepsilon n$ data variables. He can output the correct value, if Alice tells him her address variables and the decisive data variable is among Bob's variables. Otherwise, he can guess the right output with probability 1/2 (actually, he may choose always 0 as output). Then his success probability equals $2\varepsilon + \frac{1}{2}(1-2\varepsilon) = \frac{1}{2}+\varepsilon$. Hence, our approach cannot lead to substantially better results.

3 THE INNER PRODUCT FUNCTION

In this section, we prove results on the inner product function which are of the same flavor as the results on the direct storage access function in Section 2. The difference is that we can prove stronger bounds on the quality of approximation even for a larger fraction of variable orderings and larger OBDDs.

Theorem 2. *Let $0 < \delta < 1/9$. The following property holds for a fraction of at least $1 - e^{-4\delta^2 n}$ of the variable orderings π for IP_n. Each function which is at least a $(\frac{1}{2} + 2^{-\frac{1}{16}(1-9\delta)n-\frac{1}{2}})$-approximation of IP_n has a π-OBDD size which is bounded below by $2^{\delta n}$.*

Proof. First, we derive a property of random variable orderings π. It is convenient to rename the variables such that $\mathrm{IP}_n(x,y) = x_1y_1 \oplus \ldots \oplus x_{n/2}y_{n/2}$. We give the first $n/2$ variables according to π to Alice and the other ones to Bob. An index i is called a singleton if x_i is given to Alice and y_i to Bob or vice versa.

Lemma 4. *With probability at least $1 - e^{-4\delta^2 n}$, a random variable ordering leads to at least $(1-\delta)n/8$ singletons.*

Proof. Let A_x resp. A_y be the set of x-variables resp. y-variables given to Alice. Similarly, let B_x and B_y be the corresponding sets of variables given to Bob. Assume, the random ordering is generated in such a way that the positions of the x-variables are chosen first. Clearly, we have $|A_x| + |B_x| = n/2$. Denote $k = |A_x|$ and assume $|A_x| \geq n/4$. The other possibility implies $|B_x| \geq n/4$ and may be handled in a symmetric way.

Note that $|B_y| = n/2 - |B_x| = |A_x| = k$. The set B_y may be constructed by drawing k balls from an urn with k black balls and $n/2-k$ white balls corresponding to all possible indices of y-variables. The black balls correspond to indices ocurring in A_x and the white balls correspond to the indices in B_x. Clearly, the number of black balls among the k ones selected for B_y is the number of indices occurring in both A_x and B_y. Hence, the number of drawn black balls is a lower bound for the number of singletons.

Our chance of getting many black balls is minimal for the minimal value $k = n/4$. Then we have a hypergeometric distribution with mean $n/8$. It is well-known that the deviation from the mean is larger for the binomial distribution with the same success probability which is $1/2$ in our case. Hence, the probability of getting at most $(1-\delta)n/8$ singletons is bounded from above by the probability of at most $(1-\delta)n/8$ successes in $n/4$ Bernoulli trials with success probability $1/2$. Now the result follows by an application of Chernoff's bound. □

We only remark that it is even possible to obtain a lower bound of $(1-\delta)n/4$ singletons if we increase the error probability a little bit.

In the following, we try to estimate the probability that $\mathrm{IP}_n(\tilde{x}) = g(\tilde{x})$ on a random input $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_{n/2}, \tilde{y}_1, \ldots, \tilde{y}_{n/2})$, where g is a function represented by a π-OBDD of size s, assuming, we know the number of singletons determined by the ordering π.

Lemma 5. *Let π be a variable ordering, such that Alice gets among her $n/2$ variables at least t singletons. Let g be a function represented by a π-OBDD G of size s. Then,* $\Pr(\mathrm{IP}_n(\tilde{x}) = g(\tilde{x})) \leq \frac{1}{2} + s^{1/2}2^{-t/2-1/2}$.

First we show how this claim implies the theorem. Let $t = \frac{1}{8}(1-\delta)n$. Assuming $s < 2^{\delta n}$, we obtain

$$s^{1/2}2^{-t/2} < 2^{\delta n/2} \cdot 2^{-(1-\delta)n/16} = 2^{-\frac{1}{16}(1-9\delta)n}$$

and the theorem follows from Lemmas 4 and 5. □

Proof of Lemma 5. We consider the communication matrix for IP_n with respect to the partition of the variables between Alice and Bob, i.e., we have $2^{n/2}$ rows corresponding to the different input vectors for the variables of Alice and similarly $2^{n/2}$ columns. Each matrix entry is the value of IP_n on the input of the row input and the column input. If we fix all variables x_j and y_j where j is not a singleton, we obtain IP_{2t} or its negation as subfunction. Hence, the communication matrix can be partitioned to $2^t \times 2^t$-submatrices which are communication matrices for IP_{2t} or its negation. For each of these submatrices $M = (M_{ij})$, w.l.o.g. Alice is the owner of all x-variables and Bob the owner of all y-variables.

It follows by an averaging argument that there is an assignment to the variables x_j and y_j where j is not a singleton such that $\Pr(\mathrm{IP}^*_n(\tilde{x}) = g^*(\tilde{x})) \geq \Pr(\mathrm{IP}_n(\tilde{x}) = g(\tilde{x}))$ for the resulting subfunctions IP^*_n and g^* of IP_n and g resp. By the above arguments, we can assume w.l.o.g. that $\mathrm{IP}^*_n = \mathrm{IP}_{2t}$. In order to prove the lemma, it is sufficient to prove

$$\Pr(\mathrm{IP}^*_n(\tilde{x}) = g^*(\tilde{x})) \leq \frac{1}{2} + s^{1/2}2^{-t/2-1/2}.$$

Let M^* be the communication matrix of g^* and let $\rho(M, M^*)$ denote the number of entries, where M and M^* do not agree. Then

$$\Pr(\mathrm{IP}^*_n(\tilde{x}) = g^*(\tilde{x})) = 1 - \rho(M, M^*)/2^{2t}.$$

We will prove that

$$\rho(M, M^*) \geq \frac{1}{2}2^{2t} - s^{1/2}2^{3t/2-1/2}$$

which implies the lemma.

Since g^* can be represented by a π-OBDD whose size is bounded by s, it follows from the well-known relations between one-way communication complexity and π-OBDD size that the communication matrix of g^* has at most s different rows. Let $r_1, \ldots, r_s$ contain the different rows of M^*. We partition M^* to a small number of constant submatrices which intuitively implies that M and M^* are quite different. We permute the rows (i.e., renumber the input vectors of Alice) such that we have at first a_1 rows equal to r_1, then a_2 rows equal to r_2 and so on. The block of a_k equal rows can be partitioned for some b_k to an $a_k \times b_k$-matrix consisting of zeros only and an $a_k \times (2^t - b_k)$-matrix consisting of ones only. Altogether we have partitioned M^* to at most $2s$ constant submatrices whose sizes are $a_k \times b_k$ and $a_k \times (2^t - b_k)$, $1 \le k \le s$.

Now we consider the corresponding submatrices of M. It is known from Lindsey's Lemma (see, e.g., Babai, Frank, and Simon (1986)) that for each subset A of a rows of M and each subset B of b columns of M it holds that

$$\left| \sum_{i \in A,\ j \in B} (-1)^{M_{ij}} \right| \le (2^t ab)^{1/2}$$

i.e., each not too small submatrix is not constant. It follows that we have to negate at least $\frac{1}{2}(ab - (2^t ab)^{1/2})$ entries of an $a \times b$ submatrix of M to obtain a constant submatrix.

By the conclusion from Lindsey's Lemma, it follows that

$$\rho(M, M^*) \ge \sum_{1 \le k \le s} \frac{1}{2}(a_k b_k - (2^t a_k b_k)^{1/2} + a_k(2^t - b_k) - (2^t a_k (2^t - b_k))^{1/2})$$

Hence,

$$\rho(M, M^*) \ge \sum_{1 \le k \le s} \frac{1}{2}(2^t a_k - (2^t a_k)^{1/2}(b_k^{1/2} + (2^t - b_k)^{1/2})).$$

The sum of all a_k is equal to 2^t and $b_k^{1/2} + (2^t - b_k)^{1/2} \le 2^{(t+1)/2}$. Hence,

$$\rho(M, M^*) \ge \frac{1}{2} 2^{2t} - \sum_{1 \le k \le s} 2^{t-1/2} a_k^{1/2}.$$

Since $x^{1/2}$ is concave, we obtain the minimum value of the right hand side for $a_k = 2^t/s$ for all $k = 1, 2, \ldots, s$. By a routine calculation, we can derive the proposed bound on $\rho(M, M^*)$ and hence also the theorem. □

It is now easy to obtain a function which is hard to approximate by π-OBDDs for arbitrary variable ordering π. We define the function shifted inner product ShIP_n on $n/2$ x-variables, $n/2$ y-variables and $\lfloor \log n \rfloor$ z-variables. The z-variables describe an integer in binary. Then, the function $\mathrm{ShIP}_n(x, y, z)$ realizes $\mathrm{IP}_n(x^i, y)$, where i is the value of the binary vector z and x^i represents the cyclic shift of the vector of x-variables by i positions to the right.

It is easy to show that for every ordering of the variables x and y, it is possible to find a value of z such that the number of singletons is at least $n/8$. Hence, Lemma 5 implies a lower bound on the size of an approximation of ShIP_n for any variable ordering.

Corollary 1. *The function* ShIP_n *does not have an* $(\frac{1}{2} + s^{1/2} 2^{-n/16-1/2})$-*approximation in size less than* s *for any variable ordering.*

Clearly, the approximation in the theorem is very poor, unless s is exponential.

4 THE MOTIVATION FROM AND CONCLUSION FOR GENETIC PROGRAMMING

Genetic programming was introduced by Koza [12] as a heuristic approach to construct a program (in the form of an S-expression) computing a function given by examples. Let us consider genetic programming restricted to Boolean concepts like Boolean formulas, binary decision diagrams, circuits etc. This restricted form of genetic programming is closely related to the following type of minimization problems.

Assume, a model for representing Boolean functions is given. It may be a model from the list above, but also some weaker model like OBDDs, DNF formulas, decision trees etc. Moreover, a complexity measure for the given model is specified. An instance of the minimization problem is described by a set $S \subseteq \{0,1\}^n$ of inputs of a function f of n variables and the values $f(x)$ for all inputs $x \in S$. The problem is to find a function g together with its representation in the given model of complexity as small as possible and such that $g(x) = f(x)$ for all x in S.

Genetic programming is a very general type of heuristics applicable to this kind of problems which is expected to allow further progress in this area. Let us point out that in genetic programming, the usual formulation of the minimization requirement is that we look for a representation of g of complexity below a bound specified among the parameters of the run.

In the present paper, we are mostly interested in the situation, where f is a total function, which is unknown, and we only have the values of f on a set of inputs S, which is not complete, i.e. $S \neq \{0,1\}^n$. In this case, the goal is to find a total function g which agrees with f on examples from S and, moreover, yields a justifiable prediction of the values of the unknown function f on the inputs not in S. The pairs $\langle x, f(x)\rangle$ for all $x \in S$ are called *training examples* and the required function g is called a *generalization* of the training examples.

In order to get provable justification, we assume that the examples are chosen at random, independently and from the same distribution. This allows to use known results concerning the PAC-learning model, which imply that under certain assumptions, the generalization problem can be reduced to the minimization problem. More exactly, it is proved in [3] (see also [4]) that under natural assumptions, it is possible to specify a complexity bound s and a number m, which is typically larger than s, so that any function g of complexity at most s, which agrees with f on m randomly chosen independent examples, is likely to be a good approximaion of f on all inputs.

In experiments with S-expressions, for a long time, only tree representations are used. Already Koza [13] has recognized the value of graph representations and has introduced ADFs (automatically defined functions), i.e., subprograms which can be used at several places. Droste [8], [9] suggested to use OBDDs and genetic programming in order to generalize a given set of training examples. In the case of π-OBDDs for a fixed ordering, subprograms used at several places are automatically identified and merged by the reduction algorithm. If the unknown function has a small OBDD representation, then this significantly helps to find the representation. Successful experiments together with a theoretical background may be found in [8], [9], [10], [16]. A possibility to adopt an existing learning algorithm for OBDD using membership and equivalence queries to a heuristic minimization procedure for incompletely specified Boolean functions is described in [2].

Experiments with minimization for total functions using π-OBDDs for a fixed ordering π were also performed, see [21], [15].

For the models like Boolean formulas (the Boolean case of S-expressions), neural networks or even more general circuits, it is hard to compare theoretical results with experimental ones, since only very partial results are known about the complexity of these powerful models. On the other hand, OBDDs represent a practically efficient model for which strong estimates of complexity are known. Hence, they provide a good opportunity for the comparison between theoretical and experimental results.

Let us recall Occam's razor theorem from [3].

Theorem 3 ([3]). *Let H be a set of functions and f any function on the same domain. Assume, $\tilde{S}$ is a collection of m examples chosen independently from a distribution D on the domain. Then, the probability that there exists a function $g \in H$, which agrees with f on all examples in $\tilde{S}$, but $\Pr(g(\tilde{x}) = f(\tilde{x})) < 1/2 + \varepsilon$, where $\tilde{x}$ is chosen from D, is at most $|H| \left(\frac{1}{2} + \varepsilon\right)^m$.*

It is possible to strengthen this theorem using the VC dimension of H, see [4]. However, since we have no nontrivial upper bound on the VC dimension of classes corresponding to OBDDs, we use the weaker form. This is almost the same as if we use the formulation using the VC dimension and estimate the VC dimension by $\log |H|$, which is a general upper bound on the VC dimension of H.

In a typical application of this theorem following [3], the set H is the set of all functions of complexity at most s in some model. In order to apply Occam's razor theorem to OBDDs, we need an upper bound on the number of nonequivalent OBDDs of a given size s. Droste [9] used a good upper bound on this number based on a system of recurrence relations. In order to achieve a closed formula for this upper bound, we use a slightly different model, namely complete OBDDs, which test every variable in every computation. For complete OBDDs, the following bound is easy to obtain using the method of counting circuits, see e.g. [18].

Lemma 6. *The number of nonequivalent complete π-OBDDs of size s for a given π is at most $(s+2)^{2s}/s! \le (es)^s$.*

Combining Lemma 6 with Theorem 3, it is possible to justify the quality of the prediction of the unknown function f, obtained by any heuristic minimization procedure for OBDDs.

Let s be the complexity of the output g of the minimization procedure. We assume that s is a random variable depending on the random choices made by the procedure. In this situation, the estimate of the quality of the generalization g depends on the actual s achieved. Even in this situation, it is possible to achieve a good level of statistical significance, although the obtained estimate is (very) slightly worse in comparison to the situation that a bound on s is known in advance.

Definition 4. Let $\varepsilon(m, s, \delta)$ be defined by the formula

$$\frac{1}{2} + \varepsilon(m, s, \delta) = \exp\left(-\frac{(s+2)\ln(es) + \ln\frac{1}{\delta}}{m}\right).$$

Theorem 4. *Assume that a heuristic minimization procedure succeeds to find an OBDD g of size s matching an unknown function f on m independent random examples chosen from a distribution D. Then, on level δ of statistical significance, we can assume that $\Pr(g(\tilde{x}) = f(\tilde{x})) \ge \frac{1}{2} + \varepsilon(m, s, \delta)$, where $\tilde{x}$ is chosen from D.*

Proof. First, let us assume that s is fixed. Moreover, let H_s be the set of functions of OBDD complexity at most s. Using Theorem 3, we obtain that the probability that there is a function $g \in H_s$ with $\Pr(g(\tilde{x}) = f(\tilde{x})) < \frac{1}{2} + \varepsilon(m, s, \delta)$ is at most

$$(es)^s \left(\frac{1}{2} + \varepsilon(m, s, \delta)\right)^m \le \frac{\delta}{(es)^2}.$$

Using this, we can obtain a bound on the probability that there is a function of the above property for any s by taking the sum of the bounds above for all considered values of s. Since we work with complete OBDDs, it is sufficient to consider $s \geq n$. Clearly, $\sum_{s=n}^{\infty} \delta/(es)^2 < \delta$. $\square$

It is easy to verify that if $\delta > 0$ is a constant and $s \leq \alpha m/\log m$, then $\varepsilon(m, s, \delta) \geq 2^{-\alpha(1+O(1/\ln m))} - 1/2$. It follows that a nontrivial bound requires $\alpha < 1$.

In order to find a good prediction based on this theorem, it is required that the heuristic minimization procedure succeeds to find a small OBDD that agrees with all the training examples. For simplicity, we do not consider the more general situation, where we allow a difference between the OBDD and some number of the training examples. Let us call the situation that we find such an OBDD a *compression* of the set of training examples, since the size bound required to achieve a good prediction is almost exactly the bound which guarantees that the number of bits needed to represent g is less than m.

The results of the previous sections imply that DSA_n and IP_n are hard to approximate by any π-OBDD for a random π. In the next theorem, we prove, moreover, that any function f that is hard to approximate in this sense has also the following property. If we have a set of training examples for function f^π, which is obtained from the function f by an unknown permutation π of the variables, then a reasonable compression of the examples requires also to optimize the ordering of variables used to represent g. More exactly, if we choose an ordering of the variables for solving the minimization problem at random before we start the minimization process and the ordering is not modified during the process, then, with high probability, almost no compression is possible.

Theorem 5. *Let f, s, γ, ε and a distribution D on $\{0,1\}^n$ be such that the following is true: if an ordering π is chosen from the uniform distribution on all orderings, then with probability at least $1-\gamma$, every π-OBDD h of size at most s satisfies $\Pr(f(\tilde{x}) = h(\tilde{x})) \leq \frac{1}{2} + \varepsilon$, where $\tilde{x}$ is chosen from the distribution D. Let $\tilde{S}$ be a set of m independent random examples chosen from D and let π be a random ordering. Then, with probability $1 - \gamma - (es)^s \left(\frac{1}{2} + \varepsilon\right)^m$, there is no π-OBDD g of size at most s that agrees with f on all training examples in $\tilde{S}$.*

Proof. Let us call an ordering bad, if it has the property mentioned in the theorem. A random ordering is bad with probability at least $1-\gamma$. Since the examples are chosen independently on the ordering, the distribution of the examples does not change, if we condition according to the ordering. Let us estimate the conditional probability that the m examples may be expressed using a function g of π-OBDD size at most s under the condition that the ordering π is bad. Every π-OBDD of size at most s matches all the examples with probability at most $\left(\frac{1}{2} + \varepsilon\right)^m$. Multiplying this by the number of π-OBDDs of size at most s yields an upper bound on the required conditional probability. Hence, the conditional probability that the examples may not be expressed in complexity at most s is at least $1 - (es)^s \left(\frac{1}{2} + \varepsilon\right)^m$. It follows that the probability that the ordering is bad and, moreover, the examples may not be expressed in size at most s is at least $(1-\gamma)\left(1 - (es)^s \left(\frac{1}{2} + \varepsilon\right)^m\right)$. This implies the theorem. $\square$

This general result may be combined with the results of Sections 2 and 3 to obtain the following.

Corollary 2. *For every large enough n, if we take $m = n^{\Theta(1)}$ examples for DSA_n from the uniform distribution and choose a random ordering π of the variables, then with probability at least $1 - n^{-1/2}$, there is no π-OBDD of size $\frac{1}{10}m/\log m$ matching the given m training examples.*

Proof. Let $\varepsilon, \varepsilon'$ be such that $\sqrt{\ln 2}/2 < \varepsilon < \varepsilon' < 1/2^{1/10} - 1/2$. Moreover, let $\delta < \varepsilon$ be any small positive number and let $s = \frac{1}{10}m/\log m$. Using Theorem 1, we obtain for every large enough n that a random ordering π satisfies the following. With probability at least $1 - n^{-2\varepsilon^2/\ln 2}$, there is no $(\frac{1}{2}+\varepsilon')$-approximation among functions of π-OBDD complexity at most $s \le e^{n^\delta}$. Note that in our situation, $(es)^s \le 2^{m/10}$. Using also Theorem 5, with probability at least $1 - n^{-2\varepsilon^2/\ln 2} - 2^{m/10}\left(\frac{1}{2}+\varepsilon'\right)^m \ge 1 - n^{-1/2}$, there is no π-OBDD of size at most s matching the given m training examples for DSA_n. □

Corollary 3. *Let $0 < \alpha < 1$ be a constant. For every large enough n, if we take $m = n^{O(1)}$, $m \ge n$, examples for IP_n from the uniform distribution and choose a random ordering π of the variables, then with probability at least $1 - e^{-\Omega(n)}$, there is no π-OBDD of size at most $(1-\alpha)m/\log m$ matching the given m training examples.*

Proof. Let δ and ε be positive numbers such that $\delta < \frac{1}{9}$ and $\frac{1}{2} + \varepsilon < \left(\frac{1}{2}\right)^{1-\alpha}$. Moreover, let $s = (1-\alpha)m/\log m$. By Theorem 2, for every large enough n, a random ordering π satisfies the following. With probability at least $1 - e^{-4\delta^2 n}$, there is no $(\frac{1}{2}+\varepsilon)$-approximation of IP_n among functions of π-OBDD complexity at most $s \le 2^{\delta n}$. Note that $(es)^s \le 2^{(1-\alpha)m}$. Together with Theorem 5, we obtain that with probability at least $1 - e^{-4\delta^2 n} - 2^{(1-\alpha)m}\left(\frac{1}{2}+\varepsilon\right)^m = 1 - e^{-\Omega(n)}$, there is no π-OBDD of size at most s matching the given m random training examples for IP_n. □

On the other hand, for the functions DSA_n and IP_n, there are orderings, for which a good compression is possible. This suggests that including the optimization of the variable ordering into the minimization procedure often is necessary to get a good quality of the computed generalization.

References

1. L. Babai, P. Frank and J. Simon. Complexity classes in communication complexity theory. 27. FOCS, pp. 337–347, 1986.
2. A. Birkendorf and H. U. Simon. Using computational learning strategies as a tool for combinatorial optimization. Ann. Math. and Art. Intelligence 22, pp. 237–257, 1998.
3. A. Blumer, A. Ehrenfeucht, D. Haussler and M. Warmuth. Occam's Razor. Informatoion Processing Letters, No. 24, pp. 377–380, 1987.
4. A. Blumer, A. Ehrenfeucht, D. Haussler and M. Warmuth. Learnability and the Vapnik-Chervonenkis Dimension. Journal of the ACM, Vol. 36, pp. 929–965, 1989.
5. B. Bollig and I. Wegener. Improving the variable ordering of OBDDs is NP-complete. IEEE Trans. on Computers 45, pp. 993–1002, 1996.
6. R. E. Bryant. Graph-based algorithms for Boolean function manipulation. IEEE Trans. on Computers 35, pp. 677–691, 1986.
7. R. E. Bryant. Symbolic manipulation with ordered binary decision diagrams. ACM Computing Surveys 24, pp. 293–318, 1992.
8. S. Droste. Efficient genetic programming for finding good generalizing Boolean functions. Genetic Programming'97, pp. 82–87, 1997.
9. S. Droste. Genetic programming with guaranteed quality. Genetic Programming'98, pp. 54–59, 1998.
10. S. Droste and D. Wiesmann. On representation and genetic operators in evolutionary algorithms. Submitted to IEEE Trans. on Evolution Computation, 1998.
11. J. Hromkovič. *Communication Complexity and Parallel Computing.* Springer, 1997.
12. J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: The MIT Press, 1992.

13. J. Koza. *Genetic Programming II. Automatic Discovery of Reusable Programs.* MIT Press, 1994.
14. E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press, 1997.
15. H. Sakanashi, T. Higuchi, H. Iba and K. Kakazu. An approach for genetic synthesizer of binary decision diagram. IEEE Int. Conf. on Evolutionary Computation, ICEC'96, pp. 559–564, 1996.
16. T. R. Shiple, R. Hojati, A. L. Sangiovanni-Vincentelli, R. K. Brayton. Heuristic Minimization of BDDs Using Don't Cares, 31st ACM/IEEE Design Automation Conference, pp. 225–232, 1994.
17. D. Sieling, On the existence of polynomial time approximation schemes for OBDD minimization. STACS'98, LNCS 1373, pp. 205–215, 1998.
18. I. Wegener. *The Complexity of Boolean functions.* Wiley-Teubner series in computer science, 1987.
19. I. Wegener. Efficient data structures for Boolean functions. Discrete Mathematics 136, pp. 347–372, 1994.
20. I. Wegener. *Branching Programs and Binary Decision Diagrams – Theory and Applications.* To appear: SIAM Monographs on Discrete Mathematics and Applications, 1999.
21. M. Yanagiya. Efficient genetic programming based on binary decision diagrams. IEEE Int. Conf. on Evolutionary Computation ICEC'95, pp. 234–239, 1995.