The Hardness of Approximating Minima in OBDDs, FBDDs and Boolean Functions

S. A. Seshia R. E. Bryant

August 2000

CMU-CS-00-156

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

This paper presents approximation hardness results for three equivalent problems in Boolean function complexity. Consider a Boolean function f on n variables. The first problem is to minimize the level i in the Ordered Binary Decision Diagram (OBDD) for f at which the number of nodes is less than 2^{i-1}. We show that this problem is not approximable to within the factor $2^{\log^{1-\epsilon} n}$, for any $\epsilon > 0$, unless NP is contained in RQP, the class of all problems solvable in random quasi-polynomial time. This minimization problem is shown to be equivalent to the problem of finding the minimum size subset S of variables so that f has two equivalent cofactors with respect to the variables in S. Both problems are proved equivalent to the analogous problem for Free BDDs, and hence the approximation hardness result holds for all three.

This research is sponsored in part by the National Science Foundation, Award Number CCR-9805366.

The first author is supported in part by a National Defense Science and Engineering Graduate Fellowship.

Keywords: Approximation algorithms, Complexity Theory, Approximation hardness, Binary Decision Diagrams, Coding Theory, Trellises.

1 Introduction

Ordered Binary Decision Diagrams (OBDDs) are directed acyclic graphs that form canonical representations of Boolean functions [3]. The OBDD is a widely used data structure in Computer-Aided Design and Verification. The size of the OBDD depends on the order in which variables are tested. The problem of finding a variable ordering that minimizes the size of an OBDD is known to be NP-complete [2]. This problem is also known to be inapproximable to a constant factor, unless P=NP [7].

This paper takes a step further in studying the approximability of the OBDD minimization problem. We show the approximation hardness of the following related problem. Consider an OBDD for a Boolean function f on n variables. We wish to find a variable ordering that minimizes the level i at which the number of nodes is less than 2^{i-1}, the maximum possible number. The decision version of this problem has already been proved NP-complete [6].

This is an interesting problem because it indicates the first level at which the OBDD stops being a complete binary tree. For example, suppose that for an instance of this problem the answer is found to be n/c (where n is the number of variables and c is a constant greater than 1). This indicates that the size of the OBDD corresponding to that instance is exponential in n. This is one way of proving an exponential lower bound on the size of the OBDD for a Boolean function (or a family of Boolean functions). On the other hand, if the answer to this problem is a small constant, we cannot conclude much since it is still possible for the OBDD to grow exponentially beyond that level. We will refer to this problem hereafter as the "OBDD level minimization" problem.

Our proof of approximation hardness is based on the connection recently made by Lafferty and Vardy between OBDDs and minimal trellises from coding theory [6]. Their result states that the minimal proper trellis for a binary linear code C of minimum distance greater than one is essentially the same as the OBDD representing the characteristic function of the code C. It follows that for binary linear codes of minimum distance greater than one, the OBDD level minimization problem that is the subject of this paper is the same as the problem of finding the minimum level i in a minimal trellis that has less than 2^{i-1} nodes. We refer to the latter problem as the "trellis level minimization" problem.

Our work also relies on two other recent results. In a 1997 paper [8], Vardy proved the NP-completeness of the problem of finding the minimum distance of a linear code. He also outlined a polynomial-time reduction to the trellis level minimization problem from the minimum distance problem. In another paper [5], Dumer, *et al.* show that the minimum distance problem cannot be approximated to within a factor of $2^{\log^{1-\epsilon} n}$, for any $\epsilon > 0$, unless NP is contained in RQP, the class of problems solvable in random quasi-polynomial time.

The factor $2^{\log^{1-\epsilon} n}$ occurs naturally in the theory of approximation hardness, and is characteristic of a whole class of problems, as noted in the survey by Arora and Lund [1]. Moreover, the function $\psi(n) = 2^{\log^{1-\epsilon} n}$ almost equals n for small values of ϵ; e.g., for $\epsilon = 0.001$, $\psi(1000) = 984.3$. Thus, this is a far more significant factor of inapproximability than a constant factor. For the OBDD level minimization problem addressed in this paper, the size of this factor means that even if the minimum is small, say 2 or 3, we cannot find a reasonable approximate answer in polynomial time - the best we can do is n. In other words, we can derive no more information about the minimum than what we already have.

On the other hand, the assumption $NP \not\subseteq RQP$ is a stronger assumption to make than $P \neq NP$. A quasi-polynomial function is one that grows slower than $2^{\log^c n}$ for some constant c. NP $\subseteq$ RQP means that every problem in NP has a probabilistic algorithm that runs in time quasi-polynomial in the input size, and that always rejects NO instances and accepts YES instances with high probability.

As in the case of the $P = NP$? question, it is generally believed that $NP \not\subseteq RQP$.

This paper makes the following contributions. First, we show that Vardy's reduction to the trellis level minimization problem preserves approximation hardness to within a factor of $2^{\log^{1-\epsilon} n}$. This coupled with Lafferty and Vardy's result shows the approximation hardness, with the same factor, of the OBDD level minimization problem. Secondly, we show that the OBDD level minimization problem is equivalent to the problem of finding the minimum size subset S of variables so that the Boolean function (represented by the OBDD) has two equivalent cofactors with respect to the variables in S. Thus, the same approximation hardness result carries over to this problem as well. Finally, we show that the approximation hardness result also extends to a BDD variant called the Free BDD (FBDD), in which the ordering of variables can be different along different paths.

The rest of this paper is organized as follows. We define the problem and state background results in Section 2. Preservation of inapproximability is shown in Section 3. Equivalences are proved in Section 4. Finally, we make concluding remarks in Section 5.

2 Definitions and Background

2.1 Ordered Binary Decision Diagrams

An Ordered Binary Decision Diagram (OBDD) is a directed acyclic graph that forms a canonical representation of a Boolean function $f(x_1, x_2, \ldots, x_n)$. Each non-leaf node in the OBDD represents a binary test on an input variable x_i, and has two outgoing edges labeled 0 and 1, representing the result of the test. There are two leaf nodes labeled 0 and 1. All paths in the graph go from a designated root node to one of the leaf nodes. The order in which variables are tested along a path is the same for all paths. A path leading to the 0 (1) leaf node represents an assignment to input variables on which f evaluates to 0(1). There are no isomorphic subgraphs in an OBDD. Further details on BDDs can be found in the survey by Bryant [4].

We can formally define the OBDD level minimization problem as follows:

Definition 1 OBDD level-Minimization (i-BDD)

Instance: *A Boolean function $f(x_1, x_2, \ldots, x_n)$ specified in terms of a representation of size polynomial in n. Let n_j be the number of nodes at level j.*

Problem: *To find an ordering of $x_1, x_2, \ldots, x_n$ to minimize the level i of the corresponding OBDD at which $n_i < 2^{i-1}$ first holds.*

2.2 Binary Linear Codes and Minimal Trellises

A binary error-correcting code C of block length n is a collection of strings(codewords) from $\mathcal{F}_2^n = \{0,1\}^n$. The *Hamming distance* ("distance") between two strings $x, y \in \mathcal{F}_2^n$, denoted $d(x, y)$, is the number of positions in which x and y differ. The *Hamming weight* ("weight") of a codeword x is $d(x, \overline{0})$.

A binary code is linear if it is a linear subspace of $\mathcal{F}_2^n$. For such a code, the minimum distance between any two codewords is the same as the weight of the minimum weight codeword. We call a linear code of block length n, dimension k and minimum distance d, a $[n, k, d]$ linear code. An $[n, k, d]$ linear code can be seen as the row-space of a $k \times n$ *generator matrix* or the null-space of a $(n-k) \times n$ *parity-check matrix*. The code $C^\perp$ generated by the parity-check matrix of C is called the *dual code* of C. The minimum distance of $C^\perp$ is denoted by $d^\perp$.

We now define the minimum distance problem:

Definition 2 Minimum Distance (MD)
Instance: *A binary* $(n-k) \times n$ *parity-check matrix* H.
Problem: *Find a nonzero vector* $x \in \mathcal{F}_2^n$ *of minimum weight* w, *such that* $Hx^t = \mathbf{0}$.

The *minimal trellis* is a directed acyclic graph that represents a code. All edges are labeled either 0 or 1. All paths in the trellis go from a designated start node (root) to a designated end node (toor). Each path from the root to the toor corresponds to a codeword; the codeword being read off as the edge labels along the path. The *time-axis* of the trellis is the sequence in which levels (numbered 1 to n)[1] appear from root to toor. The number of nodes at each level in the minimal trellis for a binary linear code is a power of 2. Further details about trellises can be found in Vardy's survey [9].

The level minimization problem for minimal trellises is defined as follows:

Definition 3 Trellis level-Minimization (TM)
Instance: *The binary* $k \times n$ *generator matrix* G *of a binary linear code.*
Problem: *Let* T *be the corresponding minimal trellis with* n_j *nodes at level* j. *Let* $s_j = \log_2 n_j$. *We wish to find the minimum level* $i \leq n$ *of* T *(by reordering the levels), such that* $s_i < i-1$*(or equivalently,* $s_i \leq i-2$*).*

The decision version of **TM** has been proved NP-complete [6].

2.3 Approximation Algorithms and Hardness

Definition 4 *An algorithm* A *is an* α*-approximation algorithm for an optimization problem* Π *if*
1. A *runs in polynomial time*
2. A *always produces a solution which is within a factor of* α *of the value of the optimal solution.*

The factor α is called the *performance ratio*. For minimization problems, $\alpha > 1$.

For some optimization problems, approximating the optimal solution to a guaranteed factor of α can be shown to be NP-hard. Proving the NP-hardness of approximating a problem Π involves giving a special kind of reduction to Π from an NP-complete problem; the reduction must produce a gap in the value of the optimum for Π. Such a reduction is said to be a *gap-producing* reduction [1]. For example, suppose we want to prove the NP-hardness of approximating a minimization problem Π to within a factor g. We can do this by producing a gap-producing reduction from SAT to Π that maps satisfiable formulae to instances whose solution is of value at most c (for some c), and unsatisfiable formulae to instances with solutions of value at least gc (see figure 1).

To see how this works, assume Π had a polynomial time approximation algorithm A_Π that guaranteed a factor $g^* < g$. Consider an arbitrary satisfiable formula; this will map to an instance whose optimum is $\leq c$. Therefore, on this instance of Π, A_Π would return an answer x, where $x \leq g^*c < gc$. Now, we know that any unsatisfiable formula will map to an instance of Π with optimum greater than gc; furthermore, A_Π can only return an answer greater than gc for this instance. This means that we can correctly conclude in polynomial time that the formula is satisfiable, and this works for an arbitrary satisfiable formula. In other words, we can solve SAT in polynomial time, which in turn implies that P = NP! Therefore, assuming P $\neq$ NP, Π is hard to approximate to within a factor of g.

The above method of proving approximation hardness of Π uses a reduction from a known NP-complete problem. It is sometimes desirable to have a mechanism that allows us to conclude

[1]Note that levels in a trellis are typically numbered 0 to $n-1$, but we number them from 1 to n to be consistent with the convention used with OBDDs.

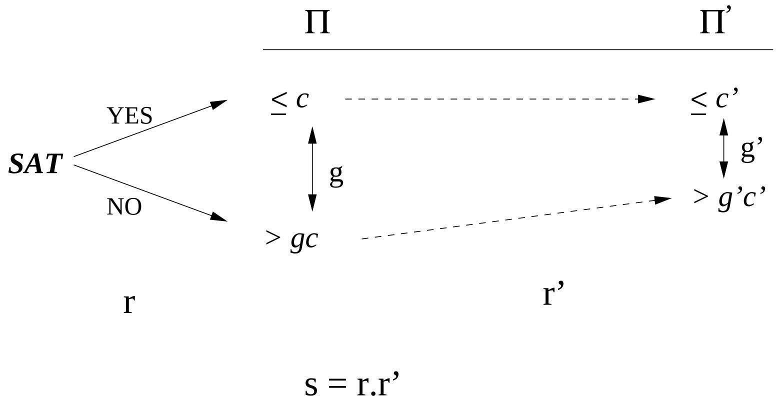

Figure 1: Gap Preserving and Producing Reductions

the approximation hardness of a problem Π', given the approximation hardness of a problem Π, instead of having to find a mapping from an NP-complete decision problem to Π'. This is achieved using the notion of a *gap-preserving* reduction.

Definition 5 Gap-Preserving Reduction: *[1] Let Π and Π' be two minimization problems. A* gap-preserving reduction *from Π to Π' with parameters $(c, g), (c', g')$ $(g, g' \geq 1)$ is a polynomial time algorithm f. For each instance I of Π, algorithm f produces an instance $I' = f(I)$ of Π'. The optima of I and I', denoted O_I and $O_{I'}$ respectively, satisfy the following two properties:*

$$O_I \leq c \quad \Rightarrow \quad O_{I'} \leq c' \tag{1}$$

$$O_I > gc \quad \Rightarrow \quad O_{I'} > g'c' \tag{2}$$

Suppose we have a gap-producing reduction r from SAT to Π. The existence of a gap-preserving reduction r' from Π to Π' proves the existence of a gap-producing reduction s from SAT to Π'; s is the composition of r and r'. In other words, the reduction s shows that achieving a performance ratio of g' for Π' is NP-hard. Figure 1 clarifies this concept.

Note however, that the gap-preserving property of a reduction does not suffice for exhibiting the ease of approximating Π given that Π' is easy to approximate.

By Lafferty and Vardy's result, we know that **i-BDD** is the same problem as **TM** for codes of minimum distance greater than one. Thus, we can extend the approximation hardness result of **MD** to one for **i-BDD** by proving that the reduction from **MD** to **TM** is gap-preserving. This proof is the subject of the next section.

3 Gap-preserving Reduction

We first reproduce Vardy's reduction from **MD** to **TM** , and then prove that the reduction is gap-preserving. Our proof relies on the following lemma, whose proof may be found in [9].

Lemma 1 Trellis Complexity and Minimum Distance: *[9] Let C be an $[n, k, d]$ linear code over $\mathcal{F}_q$ and let $d^{\perp}$ denote the minimum distance of the dual code $C^{\perp}$. Then, under all possible permutations of the time axis of the minimum trellis for C, we have,*

$$s_i = i - 1, \forall i = 2, \ldots, min(d, d^{\perp}) \tag{3}$$

Furthermore, if $i > min(d, d^{\perp})$, there exists at least one permutation of the time axis for which $s_i < i - 1$.

We reproduce Vardy's reduction of **MD** to **TM** below. The details of various code constructions in the reduction have been preserved for completeness; note however, that the only things in the reduction relevant to the proof of gap-preservation are the minimum distances of the initial and final codes.

Reduction of MD to TM: [9]

Let C be the $[n, k, d]$ binary linear code whose minimum distance d we want to determine. Given the parity-check matrix H of C, we construct a binary linear Reed-Muller code C' of length 2^m and order r, where $m = 2\lceil \log_2 n \rceil + 1$ and $r = \lceil \log_2 n \rceil$. This code C' is a $[n', k', d']$ *self-dual* code, where, $n' = 2^{2\lceil \log_2 n \rceil + 1} \leq 8n^2$, $k' = n'/2 \leq 4n^2$ and $d' = 2^{m-r} = 2^{\lceil \log_2 n \rceil + 1} \geq 2n$.

The Kronecker product construction is used to obtain a generator matrix for the product code $C^* = C^{\perp} \otimes C'$, where $C^{\perp}$ is the dual code of C. Then, the length of C^* is $n^* = nn' \leq 8n^3$, and the minimum distance $d^* = d^{\perp} d' \geq 2nd^{\perp} \geq n > d$. Finally, the dual distance of C^* is the minimum of the dual distances of $C^{\perp}$ and C'. Thus, $d^{*^{\perp}} = min(d, d') = d$. Therefore, $min(d^*, d^{*^{\perp}}) = d^{*^{\perp}} = d$.

Thus, to find d, we solve **TM** for the minimal trellis corresponding to C^*.

Proposition 1 *The reduction from* **MD** *to* **TM** *is gap-preserving.*

Proof: Let I_{MD} denote an instance of **MD** , and let I_{TM} denote the corresponding instance of **TM** obtained by the preceding reduction. Let O_{MD} denote the optimum for I_{MD} and O_{TM} that for I_{TM}.

To prove the first property of gap-preservation, let us assume $O_{MD} = d \leq c$. Let $c' = c+1$. We claim that $O_{TM} \leq c'$. To see this, consider the trellis corresponding to the code C^*, and consider i such that $d < i \leq c'$(such an i always exists by our choice of c'). By lemma 1, there must exist a permutation of the time axis such that $s_i < i - 1$. Therefore, $O_{TM} \leq c'$, and the first property holds.

Now, we prove the second property. Choose $g = 2\delta$ and $g' = \delta$ for $\delta \geq 1$. Let $O_{MD} = d > cg$. As before, $c' = c + 1$. We want to show that $O_{TM} > c'g'$, or in other words, that $s_i = i - 1$ for all $i \leq c'g'$. By lemma 1, $s_i < i - 1$ only if $i \geq d + 1 > cg + 1 = 2c\delta + 1 > c\delta + \delta = c'g'$. Hence, $O_{TM} > c'g'$, and the second property holds as well.

Thus, $O_{MD} \leq c$ implies $O_{TM} \leq c + 1$, and $O_{MD} > c(2\delta)$ implies $O_{TM} > (c + 1)\delta$. This shows that the reduction is gap-preserving. $\square$

From the approximation hardness result of Dumer, *et al.* , we know that $g = 2^{\log^{1-\epsilon} n}$. Since $g' = g/2$, ignoring constant factors in g', we can conclude that **TM** cannot be approximated to a factor of $2^{\log^{1-\epsilon} n}$, for any $\epsilon > 0$, provided NP is not contained in RQP.

Lafferty and Vardy have shown that the minimal trellis is equivalent to the ordered BDD for minimum distance $d > 1$. Thus, **TM** is equivalent to **i-BDD** for the case of $d > 1$. Since the **i-BDD** problem for the case $d > 1$ is a special case of the problem for any d, the preceding hardness result holds for **i-BDD** as well.

4 Additional Results

The approximation hardness of **i-BDD** can be used to show the approximation hardness of two related problems. The first is a restatement of **i-BDD** as a property of Boolean functions. The second is an extension of the result to a BDD-variant called the *Free BDD*. In Free BDDs (FBDDs), the variable ordering requirement is relaxed so that the variables can appear in any order along a path, but no variable can be tested more than once. Free BDDs are also known as "1-time branching programs" [10]. We define both these problems here:

Definition 6 Cofactor Equivalence for a Boolean Function (CE)
Instance: *A Boolean function* $f(x_1, x_2, \ldots, x_n)$. *Let* $V = \{x_1, x_2, \ldots, x_n\}$.
Problem: *To find a set* $S \subset V$ *of minimum size, such that there exist assignments* α, β *to variables in* S *so that* $f_\alpha = f_\beta$, *where* f_θ *denotes the cofactor of* f *with respect to the assignment* θ.

Definition 7 FBDD level-Minimization (i-FDD)
Instance: *A Boolean function* $f(x_1, x_2, \ldots, x_n)$ *specified in terms of a representation of size polynomial in* n. *Let* n_j *be the number of nodes at level* j.
Problem: *To find a FBDD representing* f *so as to minimize the level* i *at which* $n_i < 2^{i-1}$ *first holds.*

We now show the equivalence of **i-BDD** , **CE** and **i-FDD** .

Proposition 2 **i-BDD** *and* **CE** *are equivalent.*

Proof: We prove the equivalence by showing that there are $< 2^{i-1}$ nodes at level i in a OBDD for f iff there is some subset of variables S of size i, and some assignments α, β to variables in S such that $f_\alpha = f_\beta$.

For the if part, note that nodes at level i in the OBDD correspond to cofactors of f with respect to some assignment to the first i variables, and there are no duplicate nodes in an OBDD. We construct an OBDD for f with the variables in S on the top of the order, and since $\exists \alpha, \beta$ s.t. $f_\alpha = f_\beta$, the number of nodes at level i in this OBDD must be less than 2^{i-1}.

For the only if part, since there are $< 2^{i-1}$ nodes at level i, there must be two paths p_1 and p_2 from the root node to the same node N at level i. Variables in paths p_1 and p_2 are not re-tested anywhere below level i. Therefore, let S be the union of the sets of variables occurring on p_1 and p_2. Then, we can pick two assignments, α and β, over variables in S, consistent with p_1 and p_2. N is the unique node that corresponds to f_α and f_β, and so, $f_\alpha = f_\beta$. $\Box$

Proposition 3 **CE** *and* **i-FDD** *are equivalent.*

Proof: Once again, we prove the equivalence by showing that there are $< 2^{i-1}$ nodes at level i in a FBDD for f iff there is some subset of variables S of size i, and some assignments α, β to variables in S such that $f_\alpha = f_\beta$.

The if part is straightforward. Every OBDD is also a FBDD. Therefore, we construct an OBDD with the variables in S on the top of the order and the number of nodes at level i must be less than 2^{i-1}.

For the only if part, since there are $< 2^{i-1}$ nodes at level i, there must be two paths p_1 and p_2 from the root node to some node N at level i. Now consider a variable x_j that occurs on p_1 but not on p_2. We claim that x_j cannot occur on any path starting at N. This is because if it did occur on a path q starting at N, the test of x_j on q would be redundant on the path formed by concatenating p_1 with q. This redundant test will be eliminated. Thus, the path p_2 can be viewed as if it contained both the test $x_j = 0$ and $x_j = 1$.

Let S be the union of the sets of variables occurring on p_1 and p_2. Then, we can pick two assignments, α and β, over variables in S, consistent with p_1 and p_2. N is the unique node that corresponds to f_α and f_β, and so, $f_\alpha = f_\beta$. $\Box$

Since **i-FDD** and **CE** are equivalent to **i-BDD** , both problems cannot be approximated to a factor of $2^{\log^{1-\epsilon} n}$, for any $\epsilon > 0$, provided NP $\not\subseteq$ RQP.

5 Discussion

In this paper, we have used the inapproximability result for the minimum distance problem for linear codes to show the approximation hardness of the problem of minimizing the level in an OBDD at which the number of nodes first falls below the maximum possible. The inapproximability is to a factor of $2^{\log^{1-\epsilon} n}$ under the condition that NP is not contained in RQP. We have also used this result to prove the same result for FBDDs and a result stated purely in terms of Boolean functions.

As noted earlier, the factor of $2^{\log^{1-\epsilon} n}$ almost equals n for small values of ϵ. This means that even if the solution for **i-BDD** is small, say 3, we cannot find a reasonable approximate answer in polynomial time - the best we can do is n, the maximum depth of the BDD!

These problems can be linked to other interesting problems in OBDD minimization. For example, consider the problem of finding the variable ordering that minimizes the number of nodes at a specified level i in the OBDD. We formally state this problem here:

OBDD per-level minimization (l-BDD)

Instance: A Boolean function $f(x_1, x_2, \ldots, x_n)$ specified in terms of a data structure of size polynomial in n, a level number $i, 1 \leq i \leq n$.

Problem: To finding an ordering of $x_1, x_2, \ldots, x_n$ so as to minimize the number of nodes s at level i in the OBDD for $f(x_1, x_2, \ldots, x_n)$.

l-BDD can be used to solve **i-BDD** ; more precisely, we can solve an instance of **i-BDD** for a function f by solving at most n instances of **l-BDD** for f, one for each level in the OBDD, starting with the topmost level. For the instance I_j of **l-BDD** corresponding to level j in the OBDD, we test if the minimum is less than 2^{j-1}. If it is, j is the solution for the corresponding instance of **i-BDD** . If not, we attempt to solve the next instance of **l-BDD** for level $j+1$.

The preceding mapping of **i-BDD** to **l-BDD** shows that **l-BDD** is NP-complete. We can also use this mapping to prove that **l-BDD** is inapproximable to a factor of $2-\epsilon$, for any positive ϵ. To see this, consider the characteristic function ϕ of a binary linear code of minimum distance greater than 1. The number of nodes at each level in the OBDD for ϕ is a power of 2. Thus, the solution to **i-BDD** for ϕ will be a level i at which the number of nodes $n_i \leq 2^{i-2}$. If we had a $2-\epsilon$ approximation algorithm A for **l-BDD** , we could always solve this instance **i-BDD** exactly to find i, by using A to solve the first i instances of **l-BDD** . But this contradicts our result that **i-BDD** is NP-complete. Hence we cannot have a $2-\epsilon$ approximation algorithm for **l-BDD** .

However, notice that we have not used the inapproximability of **i-BDD** anywhere in this argument. This indicates that it might be possible to use the hardness result for **i-BDD** proved in this paper to show a stronger inapproximability result for **l-BDD** .

It is less evident how the inapproximability of **i-BDD** may be used to derive a strong inapproximability result for the following OBDD minimization problem.

OBDD Minimization (BDD)

Instance: A Boolean function $f(x_1, x_2, \ldots, x_n)$ specified in terms of a data structure of size polynomial in n.

Problem: Find an ordering of $x_1, x_2, \ldots, x_n$ that minimizes the number of vertices of the corresponding OBDD for $f(x_1, x_2, \ldots, x_n)$.

As mentioned earlier, **BDD** has been proved NP-complete [2], and cannot be approximated to a constant factor unless $P = NP$ [7]. Several heuristics have been suggested to solve this problem;

however, these do not guarantee any factor of approximation. Given the widespread use of OBDDs, a poly-logarithmic approximation algorithm for **BDD** would be very useful in mitigating the size blowup that is often experienced. On the other hand, an approximation hardness result for **BDD** like the one presented for **i-BDD** in this paper would indicate the futility of searching for an algorithm that works for all kinds of Boolean functions. We believe the results of this paper represent a first step in investigating these problems.

Acknowledgments

Discussions with John Lafferty were helpful in refining our paper. We also gratefully acknowledge Avrim Blum and Sergey Berezin for their suggestions.

References

[1] Sanjeev Arora and Carsten Lund. Hardness of Approximations. In *Approximation Algorithms for NP-hard problems*. PWS Publishing Company, 1996.

[2] Beate Bollig and Ingo Wegener. Improving the variable ordering of OBDDs is NP-complete. *IEEE Transactions on Computers*, 45:993–1002, September 1996.

[3] Randal Bryant. Graph-based Algorithms for Boolean Function Manipulations. *IEEE Transactions on Computers*, 35:677–691, August 1986.

[4] Randal Bryant. Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams. *ACM Computing Surveys*, 24(3):293–318, September 1992.

[5] Ilya Dumer, Daniele Micciancio, and Madhu Sudan. Hardness of approximating the minimum distance of a linear code. In *Proceedings of FOCS 1999*, pages 475–485, 1999.

[6] John Lafferty and Alexander Vardy. Ordered Binary Decision Diagrams and Minimal Trellises. *IEEE Transactions on Computers*, 48(9):971–986, September 1999.

[7] Detlef Sieling. On the Existence of Polynomial Time Approximation Schemes for OBDD Minimization. In *Proceedings of STACS'98*, pages 205–215, 1998.

[8] Alexander Vardy. The Intractability of Computing the Minimum Distance of a Code. *IEEE Transactions on Information Theory*, 43:1757–1766, November 1997.

[9] Alexander Vardy. Trellis Structure of Codes. In *Handbook of Coding Theory, Vol. II*. North-Holland Press, 1998.

[10] Ingo Wegener. On the complexity of branching programs and decision trees for clique functions. *Journal of the ACM*, 35(2):461–471, April 1988.