

Binary Decision Diagrams for Random Boolean Functions

Dissertation

zur Erlangung des akademischen Grades

DOCTOR RERUM NATURALIUM

im Fach Informatik

eingereicht an der

Mathematisch-Naturwissenschaftlichen Fakultät II
der Humboldt-Universität zu Berlin

von Dipl.-Math.

Clemens Gröpl

geboren am 13. September 1968 in Mannheim

Präsident der Humboldt-Universität zu Berlin

Prof. Dr. Dr. h. c. Hans Meyer

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät II

Prof. Dr. Bodo Krause

Gutachter:

1. Prof. Dr. Hans Jürgen Prömel, Humboldt-Universität zu Berlin
2. Prof. Dr. Ingo Wegener, Universität Dortmund
3. Prof. Dr. Anand Srivastav, Christian-Albrechts-Universität zu Kiel

Tag der mündlichen Prüfung: 3. Mai 1999

This dissertation is published electronically at: <http://dochost.rz.hu-berlin.de/>

Author's current e-mail adress: groepl@informatik.hu-berlin.de

WWW: <http://www.informatik.hu-berlin.de/~groepl/>

Abstract

Binary Decision Diagrams (BDDs) are a data structure for Boolean functions which are also known as branching programs. In *ordered* binary decision diagrams (OBDDs), the tests have to obey a fixed variable ordering. In *free* binary decision diagrams (FBDDs), each variable can be tested at most once. The efficiency of new variants of the BDD concept is usually demonstrated with spectacular (worst-case) examples. We pursue another approach and compare the representation sizes of *almost all* Boolean functions. Whereas I. Wegener proved that for ‘most’ values of n the expected OBDD size of a random Boolean function of n variables is equal to the worst-case size up to terms of lower order, we show that this is *not* the case for n within intervals of constant length around the values $n = 2^h + h$. Furthermore, *ranges of n exist for which minimal FBDDs are almost always at least a constant factor smaller than minimal OBDDs*. Our main theorems have doubly exponentially small probability bounds (in n). We also investigate the evolution of random OBDDs and their worst-case size, revealing an oscillating behaviour that explains why certain results cannot be improved in general.

Zusammenfassung

Binary Decision Diagrams (BDDs) sind eine Datenstruktur für Boolesche Funktionen, die auch unter dem Namen *branching program* bekannt ist. In *ordered* binary decision diagrams (OBDDs) müssen die Tests einer festen Variablenordnung genügen. In *free* binary decision diagrams (FBDDs) darf jede Variable höchstens einmal getestet werden. Die Effizienz neuer Varianten des BDD-Konzepts wird gewöhnlich anhand spektakulärer (worst-case) Beispiele aufgezeigt. Wir verfolgen einen anderen Ansatz und vergleichen die Darstellungsgrößen für *fast alle* Booleschen Funktionen. Während I. Wegener bewiesen hat, daß für die ‘meisten’ n die erwartete OBDD-Größe einer zufälligen Booleschen Funktion von n Variablen gleich der worst-case Größe bis auf Terme kleinerer Ordnung ist, zeigen wir daß dies *nicht* der Fall ist für n innerhalb von Intervallen konstanter Länge um die Werte $n = 2^h + h$. Ferner *gibt es Bereiche von n , in denen minimale FBDDs fast immer um mindestens einen konstanten Faktor kleiner sind als minimale OBDDs*. Unsere Hauptsätze haben doppelt exponentielle Wahrscheinlichkeitsschranken (in n). Außerdem untersuchen wir die Entwicklung zufälliger OBDDs und ihrer worst-case Größe und decken dabei ein oszillierendes Verhalten auf, das erklärt, warum gewisse Aussagen im allgemeinen nicht verstärkt werden können.

Acknowledgements

I am grateful to everybody who supported and contributed to this work: Hans Jürgen Prömel (my supervisor), Anand Srivastav, Mathias Block, Harry Preuß, Martin Skutella – as my coauthors; also to all the many other people with whom I discussed these things: Stefan Hougardy, Bernd Kreuter, Christoph Meinel, Paul Molitor, Till Nierhoff, Ralf Oelschlägel, Martin Sauerhoff, Detlef Sieling, Anusch Taraz, Thorsten Theobald, . . . ; and my wife Antje.♥

The graduate program ‘Algorithmische Diskrete Mathematik’ provided financial allowance and a high quality scientific framework. The graduate school ‘Algorithmische Diskrete Mathematik’ is supported by the Deutsche Forschungsgemeinschaft, grant GRK 219/2-97.

Contents

1	Introduction	7
1.1	Ordered Binary Decision Diagrams	10
1.2	Some Variants of OBDDs	12
1.3	The Variable Ordering Problem	14
1.4	Free Binary Decision Diagrams	15
1.5	An Invitation to Probabilistic Analysis	18
1.6	Previous Work and This Dissertation	19
1.7	The Results in More Detail	24

PART 1: RANDOM OBDDS

2	The Worst-Case Size of Quasireduced and Reduced OBDDs	31
2.1	The Function L	32
2.2	The Worst-Case Size of Quasireduced OBDDs	33
2.3	The Worst-Case Size of Reduced OBDDs	37
2.4	Remark on the History of These Bounds	38
3	The Expected Size of qOBDDs with a Fixed Variable Ordering	41
3.1	Prerequisites	42
3.2	The Expected Size of the Levels of a Random qOBDD	46
3.3	The Expected Size of Random qOBDDs with a Fixed Variable Ordering ..	53
4	The Strong Shannon Effect for qOBDDs with Optimal Variable Orderings .	55
4.1	Azuma's Inequality	55
4.2	Large Deviations from the Expected qOBDD size	56
4.3	Optimal qOBDDs for Random Boolean Functions	57
5	The Effect of the Deletion Rule	59
5.1	The Expected Reduction by the Deletion Rule	59
5.2	Large Deviations from the Expected Reduction by the Deletion Rule	62
5.3	Optimal Variable Orderings and the Deletion Rule	64
5.4	The Weak Shannon Effect for OBDDs	65
6	Comparing the Reduction Rules	67
6.1	Comparing the Expected Amount of Reduction	67
6.2	Another Kind of Large Deviation Inequalities	73
7	Other Decision Diagrams with a Variable Ordering	77
7.1	Zero-Suppressed Binary Decision Diagrams (ZBDDs)	77
7.2	Ordered Kronecker Functional Decision Diagrams (OKFDDs)	77

PART 2: RANDOM FBDDs

8	Minimal FBDDs	81
8.1	Definitions	81
8.2	Strong Reduction Rules	83
9	Quasireduced FBDDs and the Deletion Rule	87
9.1	Algorithm INVERSEDELETION	87
9.2	A Property of INVERSEDELETION	89
9.3	Probabilistic Analysis of INVERSEDELETION	91
10	FBDDs and the Merging Rule	95
10.1	Algorithm INVERSEMERGING	95
10.2	Relating qFBDDs to qOBDDs	96
10.3	The Minimal FBDD Size of Random Boolean Functions	100
10.4	Some Remarks on Main Theorem 2 (i)	102
11	Small FBDDs and large OBDDs	105
11.1	Algorithm SIMPLETYPE	105
11.2	The Expected Size of the Set U	107
11.3	Large Deviations from the Expected Size of the Set U	111
11.4	Small FBDDs via SIMPLETYPE	112
11.5	How <i>Not</i> to Improve Algorithm SIMPLETYPE	113
12	Appendix	115
12.1	Notation	115
12.2	Inequalities	115
12.3	Asymptotics	115
13	References	117
	List of Publications	125

1 Introduction

Binary Decision Diagrams are an efficient data structure for Boolean functions. Since Boolean functions (sometimes also called switching functions) are an abstraction of circuits, *Boolean manipulation* tasks arise naturally in all stages of the design process of digital circuits. For example, if a network of combinational logic has to be checked for equivalency with its specification, or an optimised version of itself, one is faced with many tasks of Boolean manipulation like *synthesis* (given representations for f and g , compute a representation for $f \otimes g$, where \otimes is a Boolean operator like \wedge , \vee , \rightarrow , \oplus , \equiv , \dots), *negation* (given f , compute $\bar{f} = 1 \oplus f$), and *equivalence testing* (decide whether the represented functions satisfy $f(\vec{x}) = g(\vec{x})$ for all \vec{x}).

A data structure problem. Let us explain with this example which difficulties arise when one has to come up with an efficient data structure for Boolean functions. There is a sort of trade-off between space requirements and algorithmic manageability. One extreme case is the combinational circuit itself, which can be viewed as a succinct representation for the Boolean function it computes. Circuits are very space efficient, because they are flexible enough to mirror many kinds of inherent structures of Boolean functions. But whereas synthesis and negation are trivial, equivalence checking is *NP*-hard for circuits. (See e.g. Papadimitriou's book [Pap94] for definitions of complexity classes.) Note that (unless $P = NP$) practical data structures for Boolean functions will always retain a certain 'heuristic flavour'; we cannot expect them to do things which are *provably* hard even for small instances. Going to the other extreme, the *truth table* is an example for a data structure that does not really care about 'structures'. It is just 'data'. While algorithms are trivial for truth tables, the space requirements are dramatic. An n -variable Boolean function has 2^n truth table entries. Typical circuits (or design blocks) have tens or hundreds of inputs, and thus a truth table approach is totally infeasible, except for the simplest cases.

We have seen from two extreme cases that every data structure for Boolean functions is a compromise between memory requirements and efficiency of manipulation algorithms, or between *space* and *time*. Basic Boolean manipulation tasks should have fast algorithms (though probably not too fast, as we have seen), but there will always be Boolean functions for which the memory requirements are high.

Counting arguments. Another reason why Boolean functions are hard to deal with algorithmically is that they exist in abundance. For n inputs, there are 2^{2^n} Boolean functions. This is because for each of the 2^n input combinations we can choose an output value. Already in one of the first papers on "The Synthesis of Two-Terminal Switching Circuits", Shannon [Sha49] proved that simply because we need a lot of basic building blocks in order to have enough choices to connect them in so many

different ways, almost all Boolean functions have a minimal circuit size of $\Omega(2^n/n)$ — and circuits are among the smallest representations we know.¹ At first sight, from these counting arguments one might be led to the conclusion that “all hope is vain”, but then numbers are not the whole truth.

Circuit complexity. Many Boolean functions which are encountered in practical applications (and in particular those which will fit on a chip of silicon) have a circuit size of $2^{o(n)}$. This observation also has a counterpart in theory. Despite many years of research on lower bounds for circuit size, we are still unable to come up with an *explicitly defined* Boolean function whose circuit complexity is only just *super-linear*. Here ‘explicitly defined’ (roughly) means that ‘tricks’ like diagonalisation are not allowed. In fact, the fifteen-year-old $3n$ lower bound by N. Blum [Blu84] is still unsurpassed. However in the meantime, more restricted computation models (e. g., monotone circuits) have been investigated, and larger (even exponential) lower bounds have been proved for them. The complexity of Boolean functions is a fascinating area of research on its own with many deep and surprising results. (See [Weg87] for an extensive monograph.)

Normal forms. Each representation of Boolean functions emphasises *some* structural aspects and (therefore) neglects others. Circuits are not only a representation for Boolean functions, but also a *model of computation* — a program that decides a certain property in fact computes a Boolean function. Computation time is an important parameter in the complexity theory of algorithms. Programs should be fast, and circuits should have small *depth*, that is the (largest) number of gates which a signal must pass from an input to the output. Minimising the depth of logic units is most critical e. g. in the design of microprocessors, in order to accelerate the clock cycle. Therefore, theoreticians have investigated symbolic representations (formulae) for Boolean functions which correspond to circuits of bounded depth. The usual canonical forms like conjunctive normal form (DNF), disjunctive normal form (DNF), and ring sum expansion (RSE, also called parity normal form (PNF) or Reed-Muller expression), which are taught in the undergraduate courses, belong to this class. These representations assign with each Boolean functions a *unique*, i. e., canonical formula, and therefore equivalence testing can be performed in polynomial time *with respect to the size of the representation*. However, simple tasks of Boolean synthesis can result in an exponential blow-up. Therefore, many synthesis tools (e. g., MINI [HCO74], ESPRESSO [BHMS84], MIS [BRSW87], and BOLD [HLJ+89]) have used a combination of non-canonical representations and heuristic manipulation techniques. We will not go into the details here.

Branching programs (or BPs, for short) emphasise the aspect that all computation over finite domains is based on making decisions. For example, if we want to compute the product of two numbers, all we have to do is to decide which digits to write down.

¹ For definitions of O , o , Ω , ω , Θ , and \sim , see Section 12. (Note the usage of absolute values.)

How does a program look like that uses nothing but decisions as its means of computation? It consists of a lot of ‘if-then-else’ statements, in combination with ‘goto’s. The program ends when an output instruction is reached. This model of computation is called a *branching program* (abbreviated BP) and has a history which is almost as old as that of circuits [Lee59].

Read-once branching programs. Branching programs are a representation for Boolean functions, whose space requirements lie in between those of circuits and formulas (up to constant factors). An early result of Cobham [Cob66] states that the BP size is closely related to the space requirements of nonuniform Turing machines. Barrington [Bar89] showed that the complexity class NC^1 contains precisely those Boolean functions which have polynomial BPs of width 5. Again, lower bounds are of great interest. A natural restriction is to allow every variable to be tested at most once [Mas76]; yet there exist powerful lower bound techniques for the size of such *read-once branching programs* (BP1s) for special functions. For example, Ponzio [Pon95] has shown that the ‘middle’ bit of integer multiplication has BP1-size $\geq 2^{\sqrt{n}/5}$. (Here, $z_{2n-1} \cdots z_0 := x_{n-1} \cdots x_0 \cdot y_{n-1} \cdots y_0$, and the ‘middle’ bit is z_n .)

Equivalence testing. The way in which branching programs achieve a concise representation is by identification of *isomorphic substructures*. If it turns out that some part of an ‘if-then-else’ program is identical to another, then we can redirect the ‘goto’s appropriately and save some lines of code. Starting from a complete binary decision tree (that sort of branching program surely exists for every Boolean function), we can identify isomorphic substructures until we end up with a acyclic directed graph in which no more reductions can be performed. — Now, surely we do *not* want to start with a complete binary decision tree; it will not fit into the memory. So in the process of Boolean manipulation (think of the introductory example of an equivalence test among combinatorial circuits), we must somehow start with the most basic functions and then combine them together step by step. But how do we test the equality of the represented functions? There can be many different ways to represent a function. Testing the equality of two functions given as read-once branching programs is a difficult problem for which only a *coRP* Algorithm is known [BCW80]. (That is a randomised algorithm which can falsely accept nonequivalent pairs with some constantly bounded error probability.) The precise complexity status of the problem is still unknown.

A word on terminology is appropriate. Theoretical work on lower bounds has shown what makes certain functions ‘hard’ for BPs or BP1s which nevertheless are ‘simple’ in another model of computation. Binary decision diagrams, on the other hand, were designed as a (hopefully) very efficient *data structure* for Boolean functions (see below). Although these are really just different ways to look at the same thing, the two communities have not yet agreed upon a common terminology. A (*general*) *BDD* is a *BP*

1.1 Ordered Binary Decision Diagrams

While read-once branching programs had been around for quite a while, it was only in the middle-eighties that Bryant came up with efficient algorithms for a special kind of read-once branching programs which he called *ordered binary decision diagrams* ([Bry86], see also [Bry92] for a survey). This approach turned out to be very successful for many Boolean functions encountered in real-world situations.

An ordered binary decision diagram (OBDD) is a read-once branching program in which the tests obey a fixed global variable ordering. For example, if the variable ordering is (x_1, \dots, x_n) , then x_i must not be tested after x_j , if $i < j$. The variable ordering implies that we can position the nodes of an OBDD (now viewed as an acyclic directed graph) in *levels*, one for each variable. Computation proceeds in the top-down direction. The order in which the variables are tested does not depend on their values. Thus, OBDDs are also called *oblivious* read-once branching programs in the BP-community. An OBDD is allowed to leave out some tests. This possibility to ‘jump across a level’ implies that there is another way to reduce the size of an OBDD: If both ‘goto’s of an ‘if-then-else’ statement point to the same line, then we can eliminate this test.

Canonicity. Bryant [Bry86] proved that the minimal OBDD for a given variable ordering is a *canonical* representation of Boolean functions. The acyclic directed graph structure implies that the equivalence test for OBDDs can be performed in polynomial time. Finding the minimal OBDD is also a feasible task. Let us explain why. We have already mentioned that there are two ways in which the size of a branching program can be reduced: identification of isomorphic subgraphs and elimination of redundant tests. The key observation is that both reductions can be performed using only local structural information. In the BDD community, these are called *reduction rules*. The *merging rule* asserts that two nodes of a BDD can be merged (i. e., identified) if they agree with respect to their successor nodes and the variables they test (see Fig. 1). The *deletion rule* asserts that a node can be removed if both outgoing edges point to the same successor node (see Fig.2).

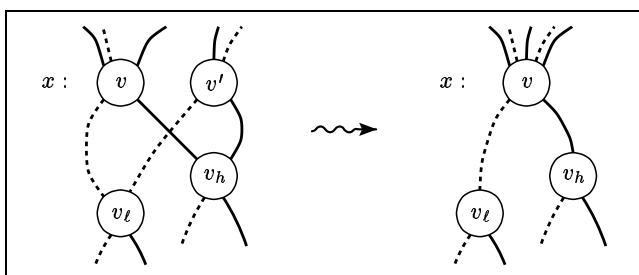


Fig. 1. Merging v and v'

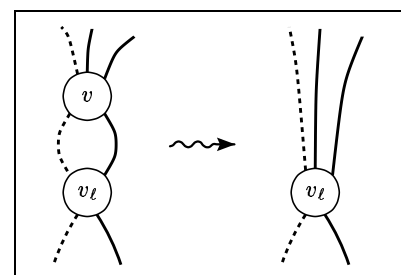


Fig. 2. Deleting v

Shared OBDDs. One can show that the variable ordering implies that an OBDD is minimal if and only if it is reduced with respect to the merging and the deletion rule. Reduced OBDDs also have the property that their nodes represent different functions. Usually, we have to deal with a lot of Boolean functions during a computation. Fortunately, OBDDs allow us to merge their representations into a single data structure (in the obvious way). This is called a *shared* OBDD. Within shared OBDDs, equivalence testing only amounts to pointer comparison, which is a *constant* time operation.

Efficient algorithms. Of course, here is not the place to describe how efficient algorithms for OBDD synthesis work, but the basic idea can be explained very quickly. Assume that a Boolean function $f = f(\vec{x}) = f(x_1, \dots, x_n)$ is represented at the ‘top’ node v of an OBDD. If the variable tested at v is x_1 , then the two successor nodes of v represent the Boolean functions

$$f_0(\vec{x}) := f(0, x_2, \dots, x_n) \quad \text{and} \quad f_1(\vec{x}) := f(1, x_2, \dots, x_n).$$

So formally, we can write f as

$$f(\vec{x}) = \bar{x}_1 \wedge f_0(\vec{x}) \vee x_1 \wedge f_1(\vec{x}). \quad (1.1)$$

This is called the *Shannon expansion* of f with respect to the variable x_1 . Now the crucial observation is that Shannon expansion *commutes* with every Boolean operator \otimes , that is, we have

$$(f \otimes g)_\alpha = f_\alpha \otimes g_\alpha \quad \text{for all } \alpha \in \{0, 1\}.$$

In this way, efficient synthesis algorithms can be designed for OBDDs, which proceed by a simultaneous depth-first traversal of the data structures.

Let us continue the success story of OBDDs with some other Boolean manipulation tasks which can be performed quite efficiently with them. We can *replace* a variable x_i by a constant $c_i \in \{0, 1\}$. This is rather clear from (1.1). (We emphasise that x_1 need not be the first variable in the ordering.) Note in contrast that if we apply this operation to a read-once branching program, it will generally change the order in which the variables are tested along some of the computation paths. Although probably not evident at first glance, this is one of the main reasons why it is so hard to work with BPIs as a data structure. We will come back to this later. If we can replace variables by constants and can perform \wedge s and \vee s, then we can also replace variables by *functions*, that is, compute $f_{g(\vec{x})}(\vec{x})$ from f and g . (For simplicity, we will not always distinguish between a function and its representation.) And of course, we can *evaluate* functions given as OBDDs. Another feasible task is *quantification*: $\exists x_1: f(\vec{x})$ and $\forall x_1: f(\vec{x})$ can be computed using replacements and synthesis. We can test for *satisfiability*, because the OBDD for a constant function is just a terminal node. More gratifying, we can even *count* and *enumerate* the *on-set* of a function f , which is defined as the set of all

assignments \vec{x} for which $f(\vec{x}) = 1$ holds. Similar for the off-set. If we test f_0 and f_1 for equivalence, then we can decide whether f depends on x_1 . There is an even simpler way to achieve this: Just look whether the OBDD has any nodes at x_i 's level.

BDD packages. Efficiency is very important in applications, and there has been put a great amount of clever work into the details of the OBDD data structure. Nowadays, OBDDs are not a thing that one would like to implement from scratch, and several *BDD-packages* are available for free, see e. g. [BRB90,Lon93,Som96,HDB]. (Since for important tasks which involve equivalence tests there are no efficient algorithms for general BDDs = BPs, the abbreviation BDD is often used as a synonym for OBDD in this context.) Recently, it has been observed that a breadth-first approach to the synthesis problem is favourable if the OBDD size exceeds the size of primary memory, due to a better cache miss rate [SRBS96,RGB97]. OBDDs with millions of nodes are not an exception in applications. See [Sen96] for an attempt to evaluate the performance of different BDD packages (which turns out to be not so easy, due to the many parameters which can have an influence). Many implementation issues are covered by the monographs [MT98,DB98].

1.2 Some Variants of OBDDs

OBDDs are the state-of-the-art data structure for Boolean functions (at least in many areas), but of course sometimes they are not as good as you want. Since the pioneering work of Bryant, many variants of OBDDs have been proposed in the literature, and there is a telling that you can have “-DD”s from A to Z. Even a leading expert in the area has to admit that it is a challenging task to keep track with the innovations [Bry95b]. Some of them might not ‘survive’ the next ten years, but others are clearly superior for certain purposes.

Zero-suppressed binary decision diagrams (ZBDDs) are a variant of OBDDs with a modified deletion rule, which allows a node to be deleted if and only if its ‘high’ successor is the terminal 0 (see Fig. 3). For example, in the situation of (1.1), the node representing f could be deleted if and only if $f_1 = 0$. Zero-suppressed BDDs were introduced by Minato [Min93]. Assume that we want to represent a system of subsets of a finite universe by its characteristic function. If the number of subsets is comparatively small, and each subset consists only of a few points, then ZBDDs are the recommended data structure. ZBDD have found applications in two-level logic minimisation [Cou94] and various combinatorial problems [Cou97,LSW95,Min96].

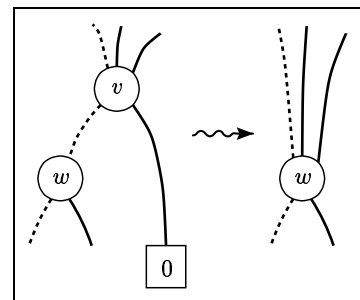


Fig. 3. The modified deletion rule, applied to v

Ordered functional decision diagrams (OFDDs) are another modification of OBDDs in which the Reed-Muller expansion takes over the part of the Shannon expansion. They were introduced by Kebschull, Schubert, and Rosenstiel in [KSR92]. Let f_0 and f_1 be as in (1.1) and $f_2 := f_0 \oplus f_1$. Then we have

$$f(\vec{x}) = f_0(\vec{x}) \oplus x_1 \wedge f_2(\vec{x}). \quad (1.2)$$

The deletion rule for OFDDs is syntactically the same as for ZBDDs, but now it has the meaning that a node can be eliminated if and only if the function it represents does not depend on the variable tested there. OFDDs are particularly useful for algorithms that deal with the ring sum expansion [TM94,DTB94], although standard operations like \wedge and \vee can lead to an exponential blow-up [BDW95].

Ordered Kronecker functional decision diagrams. There is another possibility for functional decomposition, namely

$$f(\vec{x}) = f_1(\vec{x}) \oplus \bar{x}_1 \wedge f_2(\vec{x}). \quad (1.3)$$

which leads to a data structure similar to OFDDs. Since there exist functions which have polynomial OBDD size and exponential OFDD size and vice versa, Drechsler, Sarabi, M. Theobald, Becker, and Perkowski [DST+94] combined the three decomposition types into a hybrid data structure which they called ordered Kronecker functional decision diagram (OKFDD). In OKFDDs, each variable is assigned one of the three decomposition types (1.1), (1.2), and (1.3). Still OKFDDs are a canonical representation for each such decomposition type list and variable ordering and can be manipulated efficiently (but note the remark on OFDDs above). There exist functions for which OKFDDs are exponentially smaller than both OBDDs and OFDDs [BDT95].

We will deal with ZBDDs, OFDDs and OKFDDs in Section 7.

Complementing edges. Another subtlety about decision diagrams (of all kinds) is the use of *complementing edges* [Ake78,Kar88,MB88]. A complementing edge pointing to some node which represents g is like a normal edge pointing to a node which represents \bar{g} . Although complementing edges can reduce the size of a decision diagram by at most a factor of two, they are very important in practice, because the complement flag can be stored in the lowest bit of the pointers without increasing the amount of memory allocated per node, at the price of a marginal increase on the running time.

Word-level decision diagrams. Bryant proved that OBDDs are not good at multiplications [Bry91], and this result was extended to BP1s by Ponzio [Pon95]. One of the reasons is that these decision diagrams can only deal with two-valued functions, and they are ‘blind’ for algebraic properties. Recently, BDDs have been taught some lessons in this direction [CMZ+93,BFG+93,LPV94,BC95,DBR96]. Verifying the correctness of arithmetic hardware can be very difficult, due to faults which arise only under rare

circumstances. (Even using an OBDD-based verification technique, the Intel company could have saved a lot of money [Bry95a]. But see also [Ede97] for a critical estimation of Bryant's claim.) These so-called *word-level* decision diagrams are currently a very vivid area of research. We will not be concerned with them in this dissertation.

1.3 The Variable Ordering Problem

All the decision diagrams we have seen so far have the property that the minimal representation is unique for each variable ordering (plus decomposition type list in the case of OKFDDs). But in applications, the size of the minimal representation generally depends heavily on the variable ordering. Therefore, optimisation of the variable ordering is among the most studied problems in the OBDD literature.

Complexity. From a theoretical point of view, the variable ordering problem is known to be *NP*-hard for OBDDs and OFDDs [BW96,BLSW96] and even hard to approximate to within a constant factor of the optimal size for OBDDs [Sie98b]. (The input is assumed to be given as a decision diagram in these results.)

OBDD-independent methods. Long before these negative results were proved, people developed heuristic techniques to find reasonably good variable orderings. Several heuristics for circuits have been proposed in the literature, none of which is clearly superior to the other (see e. g. [FMK88,MWBS88,MIY90]). For fanout-free circuits, optimal variable orderings can be computed in linear time [SWW96], but these are a somewhat artificial special case [PS98].

Dynamic variable ordering. OBDD-independent variable ordering heuristics have the advantage that they can be computed very fast. But usually the optimal variable ordering of the shared OBDD containing the intermediate results will change during the run of an algorithm. Fortunately, there also exist powerful variable ordering techniques which enable us to change the ordering *dynamically* (i. e., during the run time), and 'static' ordering heuristics are nowadays mainly used as initial solutions for these. The basic operation in dynamic variable ordering is the exchange of two adjacent variables in the ordering. It is not very hard to see that this so-called *swap* operation can be performed *locally*. If we have lists for the nodes on each level, then the running time of a swap is linear in the size of the two levels involved, and not in the size of the whole shared OBDD. Each swap can change the size of an OBDD by at most factor of two [BLW96].

Optimal variable orderings can be found by dynamic programming for small values of n (say, $n \leq 15$), but the running times are in general exponential [FS90,ISY91]. In many cases, a near-optimal solution will suffice, and is what we are really aiming for. Dynamic variable ordering heuristics have different ways in which they employ the swap operation in a local search strategy.

The *window permutation* heuristic [ISY91] tries all permutations of the variables in a window of constant size (typical values are $k = 2, 3$, or 4) and then restores the best. The window ‘slides’ from levels $[1 .. k]$ to $[n - k + 1 .. n]$ (or in the other direction). The window permutation heuristic is relatively fast and the results are generally better than those of OBDD independent heuristics.

Rudell’s famous *sifting* heuristic [Rud93] usually achieves much better results and has been further refined in subsequent work. For each variable in turn, it searches an optimal position in the variable ordering while keeping the relative order of the other variables unchanged. So sifting proceeds by a sequence of locally optimal *jump* operations, where each jump is performed as a sequence of swaps. Recent refinements of the sifting method deal with ‘groups’ or ‘blocks’ of variables which can be user-defined or are found during the reordering process. [PS95,PSP94,Som96,MS97]. For OKFDDs, the optimisation of the decomposition type list can be integrated into the sifting process [DB95].

Using meta-heuristics like simulated annealing, evolutionary algorithms or genetic algorithms, one can find very good variables orderings, but the running times are somewhat impractical [BLW96,DG97,MKR92]. In work not covered by this dissertation, the author was engaged in experimental studies showing the practical feasibility of a simulated annealing approach to for the (initial and dynamic) variable ordering problem [BGP+97].

1.4 Free Binary Decision Diagrams

Since the choice of a good decomposition ordering can have such a great influence on the resulting OBDD size, the variable ordering concept also has been generalised towards read-once branching programs. BP1s are called *free* binary decision diagrams (FBDDs) in the BDD community, because the outcome of a test of a variable cannot be predicted from earlier tests. (All tests are ‘free’.) In general BPs, a variable might be tested somewhere for the second time, and then the outcome is not free. Consequently, BPs can have directed source-sink paths which do not correspond to variable assignments. This is what makes the satisfiability test difficult for general branching programs. FBDDs appear as a compromise between OBDDs and BPs.

Graph orderings. In another meaning of the word, FBDDs are more ‘free’ than OBDDs, because the choice which variable to test can depend on the outcomes of earlier tests. This freedom is governed by a generalisation of the variable ordering concept which is called a (complete) *type* in the approach of Gergov and Meinel [GM94,SM93] and a *graph ordering* (formerly *oracle graph*) in the approach of Sieling and Wegener [SW94,Sie95,SW98]. A graph ordering is defined like a BP1, but has only one sink. Also, each variable is tested *exactly* once along each ‘computation’ path. An FBDD G respects a graph ordering τ if for all assignments, the variables are tested in the same order in G

and τ . Similar reduction rules as for OBDDs can be given for FBDDs. One can show that τ -FBDDs are a canonical (unique) representation of Boolean functions for every graph ordering τ . Certain restricted classes are of special interest. If the graph ordering looks like a chain, then the graph ordering concept degenerates to ordinary variable orderings. Also, tree-like graph orderings have been considered [Sie94,Sie95,BMS96].

Good news. The nice thing about FBDDs is that one can perform *synthesis* and *equivalence* checks efficiently with them. In [SW95], also a variant of the FBDD concept is investigated, which is a bit more restrictive but has better algorithmic properties. It is not necessary to go into the details here. Most important for our investigations is the fact that for every BP1 G , there exists a graph ordering τ such that G is a τ -FBDD. So the classes of Boolean functions representable by BP1s and FBDDs of a given size are identical.

The efficiency of a new decision diagram type is usually demonstrated with spectacular examples. Here is one for FBDDs. The function *hidden weighted bit* is defined as

$$\text{HWB}_n(x_1, \dots, x_n) := \begin{cases} x_{\text{wt}(\vec{x})}, & \text{if } \text{wt}(\vec{x}) := \sum_i x_i > 0; \\ 0, & \text{otherwise.} \end{cases}$$

Already this definition suggests why HWB has exponential OBDD size [Bry91]: In order to determine $\text{HWB}_n(\vec{x})$, we should know $\text{wt}(\vec{x})$; but this number is known only at the very end, and then we would still need another test to output $x_{\text{wt}(\vec{x})}$, which is not allowed. But in FBDDs, one can arrange the decomposition process in such a way that $x_{\text{wt}(\vec{x})}$ is tested near the end. The minimal FBDD size of HWB is only quadratic [GM94,SW95].

Bad news. Unfortunately, nobody knows how to find optimal or near-optimal graph orderings efficiently. Heuristics to derive tree-like graph orderings from circuit descriptions were proposed in [BMS96]. But one can show that HWB has exponential size for tree-like types [Sie95]. Very recently, Günther and Drechsler [GD99] adapted the exact minimisation algorithm of [FS90] to the FBDD case and gave a heuristic that transforms an OBDD into a smaller FBDD. Since the number of graph orderings is doubly exponential in n , FBDD minimisation algorithms have to deal with a *much* larger search space.

Another bad news about graph-ordered FBDDs is that *restricting* a variable to a constant can cause exponential blow-up. To see why, consider a function $g = g(x_2, \dots, x_n)$ which has a good (OBDD) variable ordering π and a bad variable ordering π' . Then the function $f := x_1 \wedge g(x_2, \dots, x_n)$ has a small FBDD for the tree-like type in which x_1 is tested first, followed by π' in case of $x_1 = 0$ and by π in case of $x_1 = 1$. But the function $f_1 = g$ has a large FBDD, because the top node cannot be removed by the reduction rules and so another copy of g has to be represented for the 0-branch

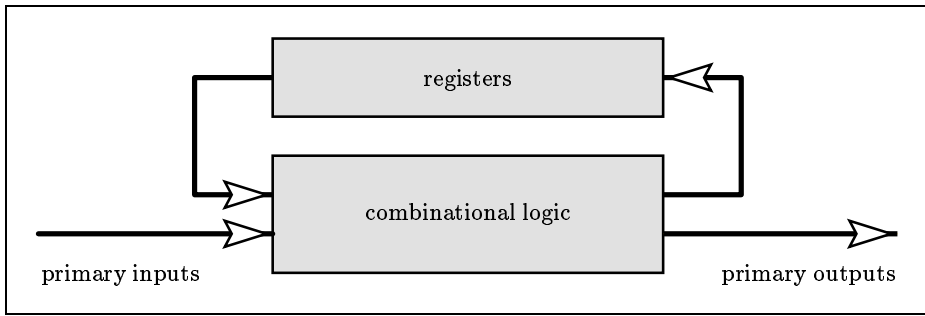


Fig. 4. A sequential circuit

of x_1 with respect to a bad variable ordering. A similar argument holds for *quantification*. — One can also show that unless we can test equivalence of BP1s in polynomial time, there is no algorithm that transforms a τ -FBDD into a τ' -FBDD in polynomial time with respect to the size of the input *and* output [Sie95].

Sequential analysis. Quantification is not necessary to test circuit equivalence by means of a canonical BP1-based data structure, since we know the representation of the zero function. Why is quantification so important? This is explained by a second major field of OBDDs applications, *sequential analysis*.

A simple abstraction of a sequential circuit is shown in Fig. 4. This is essentially a finite state machine with outputs, whose state is described by the contents of the registers. A set of states S naturally corresponds to a Boolean function (or ‘predicate’) defined by $S(\vec{x}) := 1$ iff $\vec{x} \in S$. A basic step in the analysis of sequential systems is *image computation*. Assume that the *transition relation* of the system is $T(\vec{x}, \vec{y}) := 1$ iff there exists an input such that state \vec{x} is followed by \vec{y} . Initially, the system is in a state from some set S_0 . The set of reachable states can be computed by the following recursion: $S_{i+1}(\vec{y}) := \exists \vec{x}: S_i(\vec{x}) \wedge T(\vec{x}, \vec{y})$. The least fixed point $S_i = S_{i+1}$ is the set of states reachable from S_0 . More complex specifications for sequential systems can be specified in temporal logics (e. g., computation tree logic (CTL), see [Eme90]). This approach to exact verification is called *model checking*. OBDD-based model checkers like SMV [McM93], RuleBase [BBEL96], CVE [BLPV95], or VIS [VIS96] have been applied successfully to systems with more than 10^{100} states [BCM+92, BCL+94]. The method is not restricted to hardware: communication protocols are notoriously error-prone (even published ones), and model checking has been very successful in finding some of them [McM93]. Model checking has now reached a state of maturity that begins to attract commercial interest, but the need to write down specifications in the formal notation of a temporal logic is an obstacle for its dissemination which should not be underestimated.

1.5 An Invitation to Probabilistic Analysis

We have seen that OBDDs and variants of them are very efficient in practice and thus have found applications in many areas (we only mentioned a few). But is this assertion really based on solid ground? Some trends come in and out of fashion. We need a *method* to evaluate the scope of data structures for Boolean functions.

Benchmarks. Experimental studies in VLSI design usually contain results on benchmark circuits (for references on benchmark collections see [DB98], p. 147) or ‘benchmark functions’ like HWB and the middle bit of integer multiplication. While this sometimes gives useful information for practical purposes, benchmarks are unsatisfactory for theory. For every new -DD type, one can find circuits for which the new method performs better than others, and vice versa. This can be very confusing for non-insiders. Also, no set of benchmarks covers all possible kinds of applications.

Worst-case analysis. Theoreticians have to make precise statements, and therefore they usually prove that one class is contained in another (i. e., the set of functions of a certain size) or that two classes are incomparable or that the sizes are somehow related. We mention only some of these results on the DD classes introduced above. Trivially, each OBDD and each OFDD is an OKFDD. OBDDs and OFDDs are incomparable in the sense that there exist Boolean functions whose size is polynomial in one class and exponential in the other. One can also show that some functions have small OKFDDs, but only large OBDDs or OFDDs. These relations are well understood theoretically in terms of a certain transformation that operates on Boolean functions [DB98]. The size of OBDDs and ZBDDs can differ by a factor of at most n [LSW95], which can be a lot in applications. Much less is known about the relation between OBDDs and FBDDs. The hidden weighted bit function has FBDD size $O(n^2)$ but only exponential OBDDs. However the middle bit of integer multiplication remains difficult for FBDDs.

Probabilistic analysis. Of course, FBDDs are a superclass of OBDDs, but it seems that the real question is whether there exists a significant *portion* of functions for which FBDDs are much smaller than OBDDs. Can we prove that almost all Boolean functions have a minimal FBDD which is, say, only half as large as a minimal OBDD? And what are the ‘typical’ OBDD and FBDD sizes of Boolean functions? Similar questions can be asked for OKFDDs.

What is a ‘typical’ Boolean function? A hardware engineer might say: “one that is likely to occur in our applications”. Actually, we are asking for a (more or less) realistic probability distribution on the set of all Boolean functions. It is not clear how such a probability distribution should be *defined*. Maybe it would be easier to define what is, or is supposed to be, a typical instance for an algorithm that applies OBDDs. But it is even less clear how to make this notion precise. VLSI designs are not constructed ‘at random’. Of course, artificial models can be considered: Since we want to analyse

BDDs, we could define a ‘typical OBDD’ directly, using structural properties like size and width. We indeed worked on this for a while, but due to the many side constraints, such a probability distribution is not easily amenable to mathematical analysis. Also, such a model would not say much about the ‘real world’ situations in which OBDDs will be applied. In the end, it seems that the best answer from a practical point of view is that indeed a typical Boolean function is simply one of the benchmarks.

But this dissertation is solely devoted to theoretical investigations, and from a theoretical point of view, it is clearly most natural to suppose that all Boolean functions are equally likely. That is, we will consider binary decision diagrams for random Boolean functions which are *uniformly* distributed. We do not claim that this is a ‘realistic’ model. Our choice is justified by the results we obtain.

Probabilistic analysis of algorithms and data structures is an alternative to reasoning in terms of worst-case examples and has a long history in theoretical computer science. For example, the *simplex algorithm* for linear programming has exponential worst-case running time [KM72], but the expected running time is polynomial under reasonable assumptions [Bor82]. The *quicksort* algorithm needs $\Omega(n^2)$ time in the worst case, but only $O(n \log n)$ on average [OW93, Chapter 2.2]. Sometimes polynomial-time heuristics perform very well on random instances. Long before Arora’s PTAS for the *Euclidean travelling salesman* problem [Aro98], it was known that the polynomial-time *patching* heuristic achieves an approximation ratio of $1 + O(\sqrt{n})$ with high probability [KS85]. Note that all these are deterministic algorithms. All randomness is governed by the distribution of the instances. Universal *hash* families lead (with high probability) to efficient data structures for all kinds of dictionary problems [MR95, Chapter 8].

1.6 Previous Work and This Dissertation

Liaw and Lin. Research on the OBDD representation of random Boolean functions has started with the work of Liaw and Lin [LL92,LL90]. They showed that almost all Boolean functions have a minimal OBDD size of at least $\frac{1}{2} \cdot 2^n/n$ and gave an upper bound on the worst-case size of a reduced OBDD (for some function of n variables) of $(2 + o(1))2^n/n$. Consequently, changing the variable ordering can only affect the OBDD size by a factor of at most $4 + o(1)$, with high probability. It is not surprising that most Boolean functions have large minimal OBDDs; we have already seen that a similar result holds for circuits. The theoretical observation that almost all Boolean functions are not very sensitive to variable orderings is in contrast to the experience from applications.

Quasireduced OBDDs. Liaw and Lin also noted that not using the deletion rule does not change the OBDD size of a random Boolean function by much (less than one percent for large n). It is well-known that blowing up a reduced OBDD by using the deletion rule in the reverse direction can only increase the size of a (non-constant)

OBDD by a factor of at most $O(n)$, and this bound is attained for the projection functions $f(\vec{x}) = x_i$ [LSW95]. An OBDD in which every variable is tested along every computation path and which is reduced with respect to the merging rule is called a *quasireduced* OBDD or simply a qOBDD. Besides theory, qOBDDs are of some relevance in breadth-first BDD packages [RGS97]. In the following, we always assume that OBDDs are reduced, unless stated otherwise.

Wegener. The results of Liaw and Lin were improved and generalised by Wegener [Weg94]. He proved that quotient of the size of the qOBDD and the size of the OBDD is only $1 + O(n 2^{-n/3})$ for all variable orderings, with high probability; this improves the 1% statement from [LL92]. He also showed that the *sensitivity* of a Boolean function, that is, the factor by which changing the variable ordering can alter the OBDD size, is only $1 + O(n^2 2^{-n/3})$ with high probability; improving the factor $4 + o(1)$ from [LL92]. He gave a formula for the expected qOBDD size as a sum over the expected *level* sizes, which also holds for OBDDs and optimal variable orderings up to terms of lower order. This made it possible to compute the expected size very precisely for each n , but the ‘global’ lower and upper bounds $\frac{1}{2} \cdot 2^n/n$ and $(2 + o(1))2^n/n$ from [LL92] were not addressed. In all theorems of [Weg94], “with high probability” can be replaced by “with probability $O(2^{-n/3+o(n)})$ ” (sometimes the $o(n)$ can be dropped). While this is exponential in n , it is only (roughly) the cubic root of the expected OBDD size.

The probabilistic method. An important methodological innovation in Wegener’s approach is the use of *urn experiments* or more generally, the *probabilistic method*. The probabilistic method is a powerful technique with applications in many areas of combinatorics which was pioneered by the famous mathematician Erdős who (co-)authored more than 1500 papers. Although the change from counting arguments to probabilities is only a replacement of terminology in the simplest cases, it gives us access to a whole host of deep and powerful results from probability theory while being a great alleviation for intuition. For an introduction to the probabilistic method, see the book of Alon and Spencer [AS91], which contains applications of the probabilistic method in random graph theory, number theory, Ramsey theory, geometry, coding theory, circuit complexity, and more. (See also [Spe94].)

The Shannon effect. The phenomenon that almost all Boolean functions have almost the same size for a certain kind of representation is called *weak Shannon effect*. If the weak Shannon effect holds and the expected size is almost equal to the worst-case size, that is, if almost all Boolean functions have almost the worst-case size with respect to this representation, we say that the *strong Shannon effect* holds.

Wegener [Weg94] proved that for almost all sequences of n , chosen from a certain probability distribution, the strong Shannon effect for optimal FBDDs holds with probability tending to 1 as $n \rightarrow +\infty$. This includes OBDDs as a special case. More precisely,

he showed the following: Assume that n is (itself) chosen at random from the interval $\{2^\ell, \dots, 2^{\ell+1} - 1\}$ according to the uniform distribution. Then the probability that almost all Boolean functions of n variables have a minimal FBDD size which is almost equal to the worst-case OBDD size tends to 1 as $\ell \rightarrow +\infty$.

Intuitively, Wegener's theorem asserts that for 'most' sequences of n , OBDDs and FBDDs for random Boolean functions of n variables have almost the same size, which is almost as large as it could be. So it seems natural (at least at first glance) to conjecture that Wegener's "almost all" result holds in fact for all sequences of n . However, we will show that this is not the case.

The Shannon effect for OBDDs. Our Main Theorem on OBDDs with optimal variable orderings for random Boolean functions gives an *exact criterion* to decide whether the strong Shannon effect holds for a particular sequence of n .

In order to make the main results easily accessible, the following formulations avoid some technical concepts which will be introduced in Section 3.1. The worst-case size of a qOBDD for a Boolean function of n variables is denoted by $W(n)$. Similarly, $W'(n)$ for reduced OBDDs. For a detailed discussion of the worst-case size of (quasi)reduced OBDDs and FBDDs, see Section 2. Here we only need to mention that they are $W'(n) \sim W(n) = \Theta(2^n/n)$.

Main Theorem 1. (on OBDDs with Optimal Variable Orderings)

Denote the minimal size of an OBDD for a Boolean function f by $Z_*(f)$. Let

$$B := \bigcup_{h \in \mathbb{N}} [2^h + h - d(h) .. 2^h + h + d(h)],$$

where d is specified below, and set $A := \mathbb{N} \setminus B$.

(i) If $n \rightarrow +\infty$ in such a way that $n \in A$ for some $d(h) = \omega(1)$, then

$$\Pr \left(Z_* = (1 - o(1))W \right) \sim 1,$$

i. e., the strong Shannon effect holds for the minimal OBDD (and qOBDD) size of random Boolean functions.

(ii) If $n \rightarrow +\infty$ in such a way that $n \in B$ for some $d(h) = O(1)$, then

$$\Pr \left(Z_* = (1 - \Omega(1))W \right) \sim 1,$$

and the strong Shannon effect does *not* hold for the minimal OBDD (and qOBDD) size of random Boolean functions.

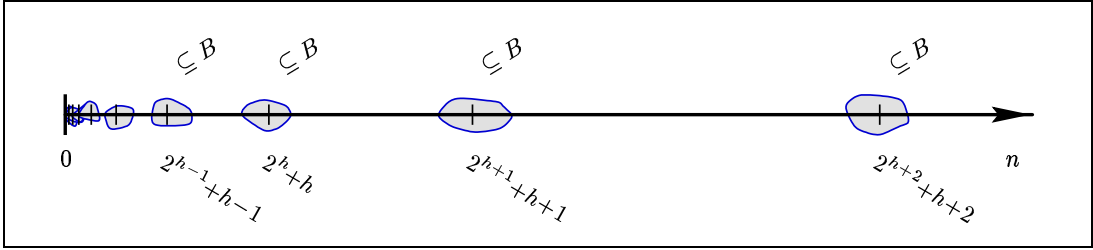


Fig. 5. The set B from Main Theorems 1 and 2

Not every sequence of n is covered by (i) or (ii) of Main Theorem 1, but every sequence not covered by (i) contains a subsequence covered by (ii), which implies that the strong Shannon effect does not hold for the original sequence.

The Shannon effect for FBDDs. When Main Theorem 1 had been proved [GPS98], we conjectured that there was an analogue for FBDDs. Indeed, the author has been able to prove that the overall picture is similar for FBDDs, but there is a remarkable difference.

Main Theorem 2. (on Minimal FBDDs)

Denote the minimal FBDD size of a Boolean function f by $\Psi(f)$. Let

$$B := \bigcup_{h \in \mathbb{N}} [2^h + h - d^-(h) .. 2^h + h + d^+(h)],$$

where d^- and d^+ are specified below, and set $A := \mathbb{N} \setminus B$.

- (i) If $n \rightarrow +\infty$ in such a way that $n \in A$ for some $d^-(h) = h^2 + \omega(1)$ and $d^+(h) = \omega(1)$, then

$$\Pr(\Psi = (1 - o(1))W) \sim 1,$$

i. e., the strong Shannon effect holds for the minimal FBDD size of random Boolean functions.

- (ii) If $n \rightarrow +\infty$ in such a way that $n \in B$ for some $d^-(h) = O(\log h)$ and $d^+(h) = O(1)$, then

$$\Pr(\Psi = (1 - \Omega(1))W) \sim 1.$$

Since the set B in Main Theorem 2 (ii) is considerably larger than in Main Theorem 1 (ii) (note that the ‘left’ interval halves have non-constant length), there is a range of n for which the minimal FBDD size is almost always a *constant factor* smaller than the minimal OBDD size. Such ‘gaps’ were already known for special functions like HWB (even of exponential size), but our result shows that this holds for *almost all* Boolean

functions, provided n is in a certain range. The upper bound on Ψ is derived from the probabilistic analysis of an algorithm which we call `SIMPLETYPE`.

Assertion (i) of Main Theorem 2 gives a range of n in which the strong Shannon effect holds for both OBDDs and FBDDs. Since we do not know the precise worst-case size of FBDDs, we cannot conclude in Assertion (ii) that the strong Shannon effect does not hold. Also, there exists a small range of n for which Main Theorem 2 makes no assertion at all (e. g., $n = 2^h$).

Investigations of the relative computation power of various decision diagram types usually focused on hierarchy results in terms of *worst-case* examples. The investigation of *expected* minimal sizes provides additional and sometimes unexpected insights. The answer sometimes depends on n .

The Shannon effect for ZBDDs and OKFDDs. Average case analysis can reveal differences which remain invisible otherwise. For example, OKFDDs are another generalisation of OBDDs which are sometimes exponentially more succinct, but we will explain in Section 7 how to prove that the minimal sizes remain almost unchanged with high probability (for all n).

The Shannon effect for general BDDs. Breitbart, Hunt, and Rosenkrantz proved that the minimal BDD size of a random Boolean function is $(1 + o(1))2^n/n$ with high probability and that the worst-case size is approximately the same (see [BHR95, page 55]). Thus the strong Shannon effect holds for the minimal BDD size and there are *no* oscillations as in our main theorems. Interestingly, their construction tests every variable at most twice. The lower bound follows from a counting argument.

We show that there are parametrisations of n for which the minimal BDD, FBDD, and OBDD size are almost the same with high probability. For other parametrisations, minimal FBDDs and OBDDs have almost the same size, but BDDs are a constant factor smaller. It is an open question whether parametrisations exist such that BDDs and FBDDs have almost the same size, but OBDDs are larger. Although we do not prove this, it seems fairly natural from our results in Section 11 that parametrisations do exist for which all the three sizes are separated by constant factors.

Evolutionary aspects. More often than one might expect, asymptotic results on OBDDs and FBDDs for random Boolean functions depend on the particular choice of the sequence of n which goes to infinity. To get a better understanding of phase transitions like those described in our main theorems, it is necessary to deal with a whole spectrum of parametrisations between the ‘extremal’ ones like $n = 2^h + h$. This ‘viewpoint of evolution’ will be emphasised in Sections 2 and 6.

The analysis of evolution processes has a prominent example in the theory of random graphs (see [Kar95] for a recent survey). For many interesting graph parameters like connectivity, hamiltonicity, planarity, the clique number, or the chromatic number

there are sharp threshold results in terms of the edge probability $p = p(n)$ known. For example, almost all graphs with $o(n)$ edges are forests, and almost all graphs with $\omega(n)$ edges consist of a unique ‘giant component’ and small components of size $O(\log n)$ with at most one cycle. Some of our methods have originated in random graph theory.

1.7 The Results in More Detail

OBDD Terminology. We collect a bunch of definitions and trivial observations in the following proposition. (See also the appendix, Section 12.)

Proposition 1.1. Let $f = f(x_1, \dots, x_n)$ be a Boolean function.

- (i) The *quasireduced ordered binary decision diagram* with the canonical variable ordering for f is denoted by $\text{qOBDD}(f)$. The nodes at level i (where x_i is tested) represent the different subfunctions of f which can be obtained by substituting the first $i - 1$ variables x_1, \dots, x_{i-1} by constants c_1, \dots, c_{i-1} . Let $Y_i(f)$ denote the number of nodes at level i of $\text{qOBDD}(f)$.
- (ii) Similarly, the *reduced ordered binary decision diagram* with the canonical variable ordering for f is denoted by $\text{OBDD}(f)$. The nodes at level i of $\text{OBDD}(f)$ represent the different subfunctions of f which can be found at level i of $\text{qOBDD}(f)$ and which depend essentially on x_i (that is, the two successor nodes are different). Let $Z_i(f)$ denote the number of nodes at level i of $\text{OBDD}(f)$.
- (iii) Clearly, Y_i is upper bounded by the growth rate of the decision tree,

$$k_i := 2^{i-1},$$

and the number of Boolean functions with $n - i + 1$ variables,

$$m_i := 2^{2^{n-i+1}}.$$

For Z_i , the upper bound m_i can be improved to

$$m'_i := m_i - m_{i+1}.$$

So

$$w_i := \min\{k_i, m_i\} \geq Y_i$$

and

$$w'_i := \min\{k_i, m'_i\} \geq Z_i.$$

- (iv) The amount of reduction achieved by the *merging rule* is

$$X_i := w_i - Y_i.$$

The additional amount of reduction achieved by the *deletion rule* is

$$X'_i := Y_i - Z_i.$$

- (v) We set $W(n) := \sum_{i=1}^n w_i$, $W'(n) := \sum_{i=1}^n w'_i$, $Y(f) := \sum_{i=1}^n Y_i(f)$, $X(f) := \sum_{i=1}^n X_i(f)$, $Z(f) := \sum_{i=1}^n Z_i(f)$, and $X'(f) := \sum_{i=1}^n X'_i(f)$. \square

The worst-case size of OBDDs and qOBDDs. There was a question in the literature [BHR95] whether the worst-case size of OBDDs could be determined up to a factor of $1 + o(1)$. We show in Section 2 that such a formula cannot be easily derived for general n , because the *relative* worst-case size $W'(n) \cdot n/2^n$ (recall that $W'(n) = \Theta(2^n/n)$) *oscillates* between $1 + o(1)$ and $2 + o(1)$. Actually, in our analysis the relative worst-case size is defined as $W/2^{L(n)}$, where L is defined in Section 2.1. The function L has the nice properties that $k_{L+1} = m_{L+1} = 2^L \sim 2^n/n$ and will be used throughout the whole dissertation. We prove estimates from a ‘local’ point of view (near $2^h + h$) and a ‘global’ point of view (between $2^h + h$ and $2^{h+1} + h + 1$). For a discussion and comparison with related work, see Section 2.4. We remark that such an oscillation does not happen for general BDDs. To the best of our knowledge, no results on the worst-case FBDD size are known except those which follow from the BDD and OBDD bounds.

The expected qOBDD size. The proof of Main Theorem 1 has three parts. In the first step, contained in Section 3, we neglect the effect of the deletion rule. Also, we consider a fixed variable ordering and do not look at large deviations. The section starts with some preliminaries which will be used throughout the whole dissertation (Section 3.1). The final result of this section is Theorem 3.13 on page 53, which the reader might want to compare with Main Theorem 1 right now. (For a quantitative result on the decrease rate of the ‘Shannon gap’, see Theorem 3.8 and Fig. 9 on page 47.)

Quasireduced OBDDs with optimal variable orderings. To account for the effect of the variable ordering in qOBDDs, we derive a strong concentration result and then “multiply with the number of variable orderings”. This is possible because the probability of exceptional Boolean functions is doubly exponential in n , which is much smaller than Wegener’s $O(2^{-n/3+o(n)})$ bound. Our large deviation result follows from Azuma’s martingale inequality, which is a standard tool of the probabilistic method. The status of the proof of Main Theorem 1 at the end of Section 4 is summarised in Theorem 4.5, page 58. This Theorem asserts (in the terminology introduced in Section 3.1) that Main Theorem 1 has been proved for qOBDDs.

The effect of the deletion rule in random qOBDDs is investigated in Section 5. We compute the expected amount of reduction by the deletion rule and derive another large deviation result using Chvátal’s inequality on the tail of the hypergeometric distribution. As a consequence, the weak Shannon effect holds for qOBDDs and OBDDs. Again we have a doubly exponentially small probability bound for large deviations (Corollary 5.6 on page 65) which improves Wegener’s exponential estimate. This completes the proof of Main Theorem 1.

Comparing the reduction rules. Going beyond Main Theorem 1, we describe the evolution of $E(X)$ and $E(X')$ in Section 6. It turns out that $n = 2^h + h$ is not the only

interesting point. Two other, more subtle phase transitions take place as n passes from $2^h + 2h - 1$ to $2^h + 2h$ and at $n = (\frac{5}{4} + o(1))2^h$. The first marks the point where the size of $E(X)$ changes from exponential to polynomial *very* suddenly, whereas $E(X')$ remains exponential. Here the ‘Shannon’ gap is minimised for qOBDDs. The second parametrisation minimises $E(X + X')$, the size of the Shannon gap for OBDDs. We take local and global viewpoints similar as in the analysis of the worst-case sizes. Since the values of $E(X)$ and $E(X')$ vary over such a wide range, the large deviation results from Sections 4 and 5, though best possible in certain cases, are rather meaningless in others. Therefore we also consider a weak, but sometimes more useful large deviation bound that has a smaller ‘cut-off’ point and follows from Markov’s inequality.

Other Decision Diagrams with a variable ordering. At the end of Part 1, we indicate how our results on OBDDs can be extended to ZBDDs and OKFDDs in Section 7. Since only minor modifications are necessary, we do not work out the details.

. ——— .

The structure of minimal FBDDs is investigated at the beginning of Part 2 in Section 8. We introduce some terminology and define *strongly* reduced FBDDs. Some easy observations are noted, which are used in the next section.

Quasireduced FBDDs. The effect of the deletion rule is estimated via the probabilistic analysis of an algorithm (INVERSEDELETION) that converts a minimal FBDD into an FBDD in which every variable is tested along every computation path (called quasireduced FBDD or qFBDD). It turns out that the minimal quasireduced FBDD size $\Psi'(f)$ is almost equal to the minimal FBDD size $\Psi(f)$ for almost all Boolean functions f (the probability bound is doubly exponential). See Corollary 9.7 on page 93.— Section 9 serves a similar purpose as Section 5. But unlike the OBDD case, the proof techniques for both parts of Main Theorem 2 are completely different.

The lower bound on the minimal qFBDD size is proved in Section 10. To show that all *levels* (these are defined similarly as for qOBDDs) are almost full with high probability, we use the results from the OBDD case and sum up the reductions for several variable orderings. This proof technique goes back to [Weg94], but here we work out the details more carefully to determine and minimise the size of the set A in Main Theorem 2 (i). Also, we have a doubly exponential probability bound, whereas Wegener’s is just $o(1)$. The main result of this section is Theorem 10.10, page 101. Finally we discuss some ideas for improvements.

The upper bound on the minimal qFBDD size follows from the (probabilistic) analysis of our new algorithm SIMPLETYPE, which is given in Section 11. Using the insights from the lower bound proof, this algorithm has been devised to collect mergings which are possible within qOBDDs for different variable orderings. The variable orderings we consider agree up to some level i^\sharp . Then we choose a set of variables which are tested

at the next few levels in such a way that at level $i' = i^\sharp + O(1)$, the mergings for all the corresponding variable orderings sum up. Again, we derive a doubly exponential probability bound via Azuma's inequality in Theorem 11.7, page 112. In some sense, our analysis of SIMPLETYPE is best possible (Section 11.5). But since the number of 'amalgamated' variable orderings is very limited, a gap remains between Assertions (i) and (ii) of Main Theorem 2.

PART 1:

RANDOM OBDDs

2 The Worst-Case Size of Quasireduced and Reduced OBDDs

In this section we investigate the behaviour of the worst-case bounds $W(n)$ and $W'(n)$ for different parametrisations of n . This is best explained in terms of the ratios $W/2^L$ and $W'/2^L$, called *relative worst-case size* henceforth. (The function $L = L(n)$ will be defined soon. We have $2^L \sim 2^n/n$.) We describe the oscillation of the quasireduced relative worst-case size and give tight lower and upper bounds, improving upon earlier results of Liaw and Lin [LL92], Heap and Mercer [HM94], and Breitbart, Hunt, and Rosenkrantz [BHR95]. The results carry over to reduced OBDDs without major changes. Finally, we discuss the history of these bounds.

But first, let us see that the upper bounds w_i and w'_i are tight. The following lemma was proved in [HM94, Lemma 2].

Lemma 2.1. For all n , there exist Boolean functions f and f' of n variables such that

$$\forall i \in [n]: Y_i(f) = w_i \quad \text{and} \quad \forall i \in [n]: Z_i(f') = w'_i.$$

Proof. We show how to construct reduced and quasireduced OBDDs matching the upper bounds, thus defining f and f' . Let i be the largest index such that $k_i \leq m_i$. For the top part of the OBDD, we take a decision tree with k_i terminals. Since $k_i \leq m_i$, we can assign different subfunctions depending on $n - i + 1$ variables to all the terminals of the decision tree. So we can guarantee that in the resulting OBDD, no mergings are possible *above* and *at* level i .

But how can we enforce that the upper bound is attained at the levels *below* i ? We need to specify the assignment of subfunctions to the leaf nodes of the top part tree in more detail. Think of a ‘universal’ shared OBDD U representing all Boolean functions of the lower variables x_i, \dots, x_n . In the reduced OBDD case, U has m_i nodes, including the sinks, whereas in the quasireduced OBDD case, the top level of U where x_i is tested (possibly redundantly) alone already has m_i nodes. In the reduced OBDD case, the terminals of the top part tree are replaced with nodes from the whole lower part, while in the quasireduced OBDD case, we only use the nodes at the top level of the lower part.

Since $2k_i = k_{i+1} > m_{i+1}$, we can choose the first $m_{i+1}/2$ subfunctions in such a way that the high- and low-successors of these nodes already ‘cover’ the second level of U (where x_{i+1} is tested) completely. The levels below $i + 1$ are ‘full’, because each node at level $j \geq i + 2$ has incoming edges from level $j - 1$. \square

2.1 The Function L

It is desirable to have a concise notation for the “critical point” i where the two upper bounds k_i and m_i meet. For technical reasons, we consider m_i instead of m'_i , although our interest is mainly in reduced OBDDs. But the difference is only marginal.

Proposition 2.2. Define the function L by the functional equation

$$L(n) + \log L(n) = n, \quad (2.1)$$

and set

$$i_\delta := L(n) + \delta + 1. \quad (2.2)$$

Then by definition,

$$k_{i_\delta} = 2^{\delta+L} \quad \text{and} \quad m_{i_\delta} = 2^{2^{-\delta}L}. \quad (2.3)$$

In particular,

$$w_{i_0} = k_{i_0} = m_{i_0} = 2^{L(n)}.$$

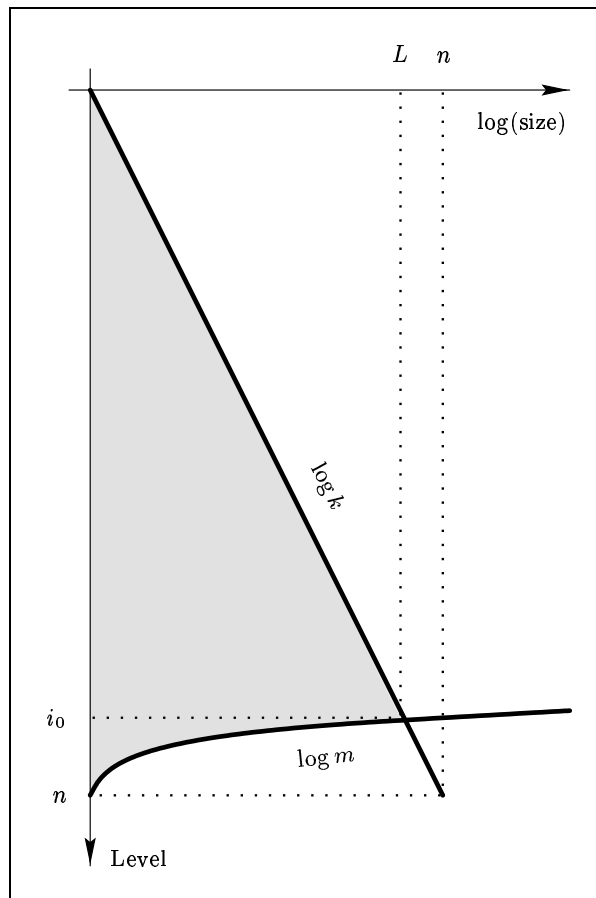


Fig. 6. The worst-case shape of a qOBDD (worst-case OBDDs look almost the same)

Note that i_0 is not an integer in general. It only marks the borderline where the worst-case level width turns from growing exponentially to shrinking doubly exponentially (see Fig. 6).

The definition of L is really important for our work. Earlier investigations dealt with L_1 [Weg94] in some way, or approximated L by other means [LL92]. In [HM94], $\lceil i_0 \rceil$ was implicit in the form of $\min\{i \in \mathbb{N} \mid k_i \geq m_i\}$, but no asymptotic for $\lceil i_0 \rceil - i_0$ was given. In our approach, we work with L itself and use the functional equation (2.1) frequently. With some habituation, this makes the computations much easier.

Unfortunately, there is no closed formula for L , whereas by the defining functional equation (2.1), the inverse function of L is simply $L^{-1}(i) = i + \log i$. As an alternative definition, one can also obtain L as a pointwise limit of a sequence of functions L_r , $r \in \mathbb{N}_0$, defined by $L_{-1}(n) := 1$ and $L_{r+1}(n) := n - \log L_r(n)$. It is elementary to show that the pointwise limit $L(n) := \lim_{r \rightarrow \infty} L_r(n)$ exists iff $n \geq 1$. In fact, the inequalities

$$L_{2r}(n) > L_{2r+2}(n) > L(n) > L_{2r+3}(n) > L_{2r+1}(n) \quad (2.4)$$

are valid for $r \in \mathbb{N}_0$ and $n > 1$. The first approximations are

$$\begin{aligned} L_0(n) &= n, \\ L_1(n) &= n - \log n, \\ L_2(n) &= n - \log(n - \log n), \\ L_3(n) &= n - \log(n - \log(n - \log n)). \end{aligned} \quad (2.5)$$

From these one can easily show that $L(n) \sim n$ and $2^{L(n)} \sim 2^n/n$. Also, $L_1(n) - L(n) = o(1)$, since $L_1(n) - L_2(n) = o(1)$. The inequalities (2.4) will be applied occasionally, but the asymptotics are more important. Again, we refer to the appendix.

2.2 The Worst-Case Size of Quasireduced OBDDs

By definition of the function L , we have

$$w_i = \begin{cases} k_i, & i \leq i_0; \\ m_i, & i \geq i_0. \end{cases} \quad (2.6)$$

To estimate W , we write out the sum and substitute (2.6).

$$\begin{aligned} W(n) &= \sum_{i=1}^{\lceil i_0 \rceil - 1} 2^{i-1} + \sum_{i=\lceil i_0 \rceil}^n 2^{2^{n-i+1}} \\ &= \sum_{i=0}^{\lceil L \rceil - 1} 2^i + \sum_{i=1}^{n-\lceil L \rceil} 2^{2^i} \\ &= 2^{\lceil L \rceil} + 2^{2^{n-\lceil L \rceil}} + \left| O\left(2^{2^{n-\lceil L \rceil - 1}}\right) \right|. \end{aligned} \quad (2.7)$$

Both leading terms are roughly of size 2^L . The exact value of $\lceil L \rceil$ is given in the next lemma.

Lemma 2.3.

$$\lceil L(2^h + h + a) \rceil = \begin{cases} 2^h + a + 1, & a \in [-2^{h-1} - 1 .. -1] ; \\ 2^h + a, & a \in [0 .. 2^h] . \end{cases}$$

Proof. Using the functional equation (2.1), we can write

$$L(2^h + h + a) = 2^h + h + a - \log L(2^h + h + a) .$$

Since $L(2^h + h) = 2^h$ by (2.1) and $\log L$ is a strictly isotone function, we have for $a \in [-2^{h-1} - 1 .. -1]$,

$$h - 1 = \log L(2^{h-1} + h - 1) \leq \log L(2^h + h + a) < \log L(2^h + h) = h ,$$

and for $a \in [0 .. 2^h]$,

$$h = \log L(2^h + h) \leq \log L(2^h + h + a) < \log L(2^{h+1} + h + 1) = h + 1 .$$

Thus, the points where $\lceil L \rceil$ does not increase are $\lceil L(2^h + h - 1) \rceil = \lceil L(2^h + h) \rceil$. From these observations, the lemma is easily inferred. \square

The next theorem gives the asymptotic value of $W/2^L$ for parametrisations of n ‘close’ to $2^h + h$. (See Fig. 7 on page 36.)

Theorem 2.4. Assume that $n \rightarrow +\infty$ is of the form $n = 2^h + h + a$, where $a = o(2^h)$. Then

$$\frac{W(n)}{2^{L(n)}} \sim \begin{cases} 2, & a \leq 0 ; \\ 1 + 2^{-a}, & a \geq 0 . \end{cases}$$

Proof. We approximate W and 2^L separately and then consider their ratio. — To estimate $W(n)$, we use (2.7) and apply Lemma 2.3. For $a \in [-2^{h-1} - 1 .. -1]$,

$$W(2^h + h + a) = 2^{2^h+a+1} + 2^{2^{h-1}} + O(2^{2^{h-2}}) \sim 2^{2^h+a+1} + 2^{2^{h-1}}, \quad (2.8)$$

and for $a \in [0 .. 2^h]$,

$$W(2^h + h + a) = 2^{2^h+a} + 2^{2^h} + O(2^{2^{h-1}}) \sim (2^a + 1) 2^{2^h} . \quad (2.9)$$

Since $L(n) \sim n \sim 2^h$, expanding L twice using the functional equation (2.1) yields

$$2^{L(n)} = \frac{2^n}{n - \log L(n)} \sim \frac{2^{2^h+h+a}}{2^h+a} \sim 2^{2^h+a}. \quad (2.10)$$

This leads to the asserted formulae for $W/2^L$: For $a \in [-2^{h-1}-1 \dots -1]$,

$$\frac{W}{2^L} \sim \frac{2^{2^h+a+1} + 2^{2^h-1}}{2^{2^h+a}} = 2 + 2^{-2^h-1-a} \sim 2$$

by (2.8) and (2.10). For $a \in [0 \dots 2^h]$,

$$\frac{W}{2^L} \sim \frac{(2^a + 1) 2^{2^h}}{2^{2^h+a}} = 1 + 2^{-a}$$

by (2.9) and (2.10). \square

Theorem 2.4 is complemented by Theorem 2.5, which describes how $W/2^L$ develops between $2^h + h$ and $2^{h+1} + h + 1$. (See Fig. 8.)

Theorem 2.5. Assume that $n \rightarrow +\infty$ and write $n = b2^h + h$ with $b \in [1, 2]$, $h \in \mathbb{N}$. Then

$$\frac{W(n)}{2^{L(n)}} = b \left(1 + 2^{(1-b)2^h} + O\left(2^{\left(\frac{1}{2}-b\right)2^h}\right) \right) \left(1 - \frac{\log b + O(h/n)}{b2^h} \right),$$

which is $\sim b$, if b converges to a real number in $]1, 2]$.

Proof. It is easy to see that h and b are well-defined for every n . The theorem is proved in a similar way as the case $a \geq 0$ in Theorem 2.4, but we must estimate $2^{L(n)}$ more accurately. First, observe that $L(n) = n + O(\log n) = b2^h(1 + O(h/n))$, so from $\log(1+x) \sim x \log e$ for $x \rightarrow 0$ we get

$$\log L(n) = \log(b2^h(1 + O(h/n))) = \log b + h + O(h/n).$$

Therefore, a refined version of (2.10) is

$$2^{L(n)} = \frac{2^{b2^h+h}}{b2^h - \log b + O(h/n)}. \quad (2.10')$$

Using (2.9) and (2.10'), we see that

$$\begin{aligned} \frac{W(n)}{2^{L(n)}} &= \frac{2^{b2^h} + 2^{2^h} + O(2^{2^h-1})}{2^{b2^h}} \cdot \frac{b2^h - \log b + O(h/n)}{2^h} \\ &= b \left(1 + 2^{(1-b)2^h} + O\left(2^{\left(\frac{1}{2}-b\right)2^h}\right) \right) \left(1 - \frac{\log b + O(h/n)}{b2^h} \right). \end{aligned} \quad \square$$

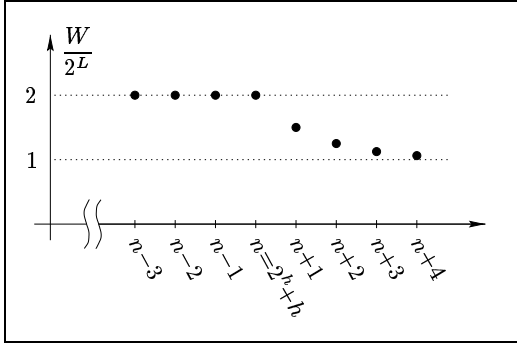


Fig. 7. The relative worst-case size of OBDDs near $n = 2^h + h$

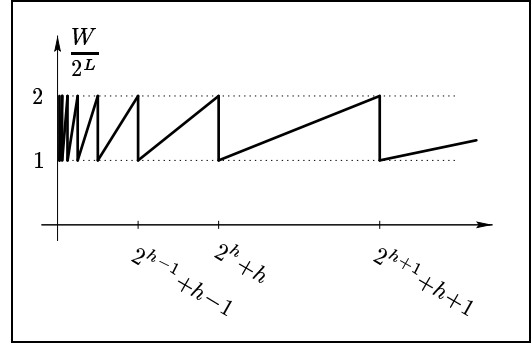


Fig. 8. The oscillation of the relative worst-case size of OBDDs

Our refined analysis of the oscillation of the relative worst-case size leads to some improvements over the upper bound of Liaw and Lin [LL92] and Heap and Mercer [HM94] and the lower bound of Breitbart, Hunt, and Rosenkrantz [BHR95]. (Essentially, the lower bound from [BHR95] is $1 + o(1)$, and the upper bound from [LL92] is $2 + o(1)$.) Our bounds are attained for certain sequences of n which are described in the proof.

Theorem 2.6. For $n \rightarrow +\infty$,

$$1 < \frac{W(n)}{2^{L(n)}} \leq 2 + O(2^{-L(n)/2}) = 2 + O\left(\sqrt{\frac{n}{2^n}}\right).$$

Furthermore, if $n \in \mathbb{N} \setminus \{2^h + h \mid h \in \mathbb{N}\}$ is large enough, then the upper bound can be improved to $W/2^L < 2$.

Proof. The lower bound holds because $W > 2^{\lceil L \rceil} \geq 2^L$ by (2.7).—For the upper bound, we use the asymptotic from Theorem 2.5. Let h and b be as defined there. First, we simplify the $O(2^{(1/2-b)2^h})$ term. Observe that

$$L(n) = L(b2^h + h) \leq L(b2^h + \log b + h) = L(b2^h + \log(b2^h)) = b2^h$$

by the isotonicity and the defining functional equation of L . Therefore,

$$\left(\frac{1}{2} - b\right) 2^h \leq \frac{-b}{2} 2^h \leq \frac{-L(n)}{2}.$$

This proves

$$2^{(\frac{1}{2}-b)2^h} = O(2^{-L(n)/2}).$$

With $a := (b-1)2^h$ (so $n = 2^h + h + a$) the asymptotic from Theorem 2.5 becomes

$$\frac{W(n)}{2^{L(n)}} = \left(1 + \frac{a}{2^h}\right) \left(1 + 2^{-a} + O(2^{-L(n)/2})\right) \left(1 - \frac{\log(1 + a/2^h) + o(1)}{a + 2^h}\right). \quad (2.11)$$

In particular, for any sequence of the form $n = 2^h + h + a$, where $a \rightarrow +\infty$ and $a = o(2^h)$, we get $W/2^L = 1 + o(1)$, which proves that the lower bound is asymptotically tight.

On the other hand, if $n = 2^h + h$, then $L(n) = 2^h$ by Lemma 2.3, and from (2.7) we can easily read off that

$$W(2^h + h) = 2 \cdot 2^{2^h} + 2^{2^h-1} + O(2^{2^h-2}) = 2L(n) + \Theta(2^{L/2}),$$

which shows that the upper bound is attained for $a = 0$. (Recall that $L(n) = n - \log n + o(1)$, so $2^L \sim 2^n/n$.)

It remains to show that $W(n)/2^{L(n)} < 2$, if $n = 2^h + h + a$ and $a \in [1..2^h]$, provided h is bigger than some fixed constant (which will not be determined here). In the following estimations, we always assume that h is large enough. Note that $L(n) = \Theta(2^h)$. If $a = 1$, then $W/2^L = \frac{3}{2} + o(1) < 2$ by (2.11). If $2 \leq a \leq \frac{2}{5}2^h$, then

$$\left(1 + \underbrace{a/2^h}_{\leq 2/5}\right) \left(1 + \underbrace{2^{-a}}_{\leq 1/4}\right) \leq \frac{7}{5} \cdot \frac{5}{4} = \frac{7}{4},$$

which implies $W/2^L \leq \frac{7}{4} + o(1) < 2$ by (2.11). Finally, if $\frac{2}{5}2^h < a \leq 2^h$, then

$$\frac{W(n)}{2^{L(n)}} = \underbrace{\left(1 + a/2^h\right)}_{\leq 2} \left(1 + \underbrace{2^{-a}}_{\leq 2^{-\frac{2}{5}2^h}} + \underbrace{O(2^{-L(n)/2})}_{2^{-\Omega(2^h)}}\right) \left(1 - \frac{\log(1 + a/2^h) + o(1)}{\underbrace{a + 2^h}_{\geq (\log \frac{7}{5} + o(1))/2^{h+1}}}\right) < 2$$

by (2.11). □

2.3 The Worst-Case Size of Reduced OBDDs

We point out how the results on $W/2^L$ from the previous section carry over to $W'/2^L$. By (2.6), $j = \lceil i_0 \rceil$ is the smallest index such that $k_j \geq m_j$. Since both k_j and m_j are powers of 2 and $m'_j = m_j(1 - |o(1)|)$ for $j = n - \omega(1)$, it follows that $j = \lceil i_0 \rceil$ is also the smallest index such that $k_j \geq m'_j$. The starting point of our analysis was (2.7).

For reduced OBDDs, we have

$$\begin{aligned}
W'(n) &= \sum_{i=1}^{\lceil i_0 \rceil - 1} 2^{i-1} + \sum_{i=\lceil i_0 \rceil}^n \left(2^{2^{n-i+1}} - 2^{2^{n-i}} \right) \\
&= \sum_{i=0}^{\lceil L \rceil - 1} 2^i + \sum_{i=1}^{n - \lceil L \rceil} \left(2^{2^i} - 2^{2^{i-1}} \right) \\
&= 2^{\lceil L \rceil} + 2^{2^{n - \lceil L \rceil}} - 3.
\end{aligned} \tag{2.12}$$

Corollary 2.7. Theorems 2.4 and 2.5 hold for $W'/2^L$ as well.

Proof. All we need to check is that since $-3 = O(2^{2^L-2})$, the estimates (2.8) and (2.9) from the proof of Theorem 2.4 are valid for W' , too. \square

The overall bounds on the relative worst-case size have a particularly nice form for reduced OBDDs.

Corollary 2.8. For large enough n ,

$$1 < \frac{W'(n)}{2^{L(n)}} < 2,$$

and both bounds are asymptotically tight.

Proof. The lower bound follows since $W' > 2^{\lceil L \rceil} \geq 2^L$ by (2.12).— Since $W' \leq W$, the upper bound follows from Theorem 2.6 except for the case $n = 2^h + h$. If $n = 2^h + h$, then $\lceil L \rceil = 2^h$ by Lemma 2.3, and we have $W' = 2^L + 2^{2^{n-L}} - 3 = 2 \cdot 2^L - 3$ by (2.12). This also shows that the upper bound is asymptotically attained. \square

2.4 Remark on the History of These Bounds

Already in his seminal paper from 1959, Lee [Lee59] proved that the worst-case OBDD size is at most $4 \cdot 2^n/n - 2$ and that Boolean functions f exist having a minimal BDD size of at least $\frac{1}{2} \cdot 2^n/n + 1$, which is also a lower bound for the OBDD case. The lower bound uses a variant of Shannon's classical counting argument [Sha49].

Liaw and Lin [LL92] improved the upper bound to $(2 + |o(1)|)2^n/n$ and showed by a family of counterexamples that this cannot be improved to $2 \cdot 2^n/n$. Heap and Mercer [HM94] showed that the worst-case size is exactly $W'(n)$, see our Lemma 2.1.

Lee's lower bound was improved by Breitbart, Hunt, and Rosenkrantz [BHR95], who showed that the worst-case BDD size is $(1 + o(1))2^n/n$. Independently of [LL92] and [HM94], they also discovered the upper bound $(2 + o(1))2^n/n$ for OBDDs. They remark that the construction from their proof (which is essentially identical to that of [LL92, HM94]) gives an upper bound of $(1 + o(1))2^n/n$ for certain sequences of n .

To the best of our knowledge, the 'evolution' of the worst-case size has not been investigated thoroughly before. Heap and Mercer gave a diagram of $Wn/2^n$ for $n \leq 160$ which clearly reveals the oscillations, but did not comment on this fact. Although [HM94, Figure 1] indicates that $(1 + o(1))2^n/n$ is the right global lower bound, they only proved $W'(n) \geq \frac{1}{2} \cdot 2^n/n$. Breitbart, Hunt, and Rosenkrantz (falsely) conjectured that $W'(n) = (1 + o(1))2^n/n$ similar to their result on BDDs, see the comments following [BHR95, Theorem 2].

Constructions like that of Lemma 2.1 are frequent in the literature, and we believe it is good advice that these should be 'tested' for a number of 'odd' parametrisations of n in a routine way in order to get an idea of the evolution.

We feel that the main (little) achievements of our approach to the worst-case size of OBDDs are:

- (1) Using 2^L instead of $2^n/n$ makes it possible to remove the $o(1)$ terms from the upper and lower bounds (Corollary 2.8).
- (2) A thorough investigation of the evolution of $W'(n)$ reveals an interesting oscillating behaviour that explains why the global upper and lower bounds cannot be improved (Theorems 2.4 and 2.5).
- (3) For any given sequence of n , we know $W'(n)$ up to a factor of $1 + o(1)$.

Actually, it was the strange 'peak' of $W(n)$ near $n = 2^h + h$ which led us to the conjecture of Main Theorem 1 (ii). (See also Proposition 3.2.)

Since the worst-case size of FBDDs lies between that of general BDDs and that of OBDDs, it is known by the above discussion up to a factor of $2 + o(1)$, and only for certain sequences of n (those for which $W'(n) \sim 2^L$) up to $1 + o(1)$. As a direction of future research, we suggest that the worst-case size of FBDDs should be determined up to a factor of $1 + o(1)$. Such a bound seems to be necessary in order to prove that the strong Shannon effect does not hold in a situation similar to that of Main Theorem 2 (ii).

3 The Expected Size of qOBDDs with a Fixed Variable Ordering

In a typical qOBDD, we would expect that not all levels are as large as in the worst-case examples. Recall that $X_i = w_i - Y_i$ was defined as the number of ‘missing’ nodes at the i -th level of a qOBDD. For each level, the expected value $E(X_i)$ can be computed by the following urn experiment [KSC78,Weg94].

Proposition 3.1. Think of the subfunctions that result from constant assignments to the first $i - 1$ variables as *balls* and of the possible subfunctions at level i as *urns*. Then at level i , there are k_i balls being thrown into m_i urns, so the expected number of non-empty urns is

$$E(Y_i) = \sum_{j \in [m_i]} \Pr(j\text{-th urn is non-empty}) = m_i (1 - q_i),$$

where

$$q_i := \Pr(\text{first urn is empty}) = \left(1 - \frac{1}{m_i}\right)^{k_i}.$$

With (2.6), it follows that

$$E(X_i) = \begin{cases} k_i - m_i (1 - q_i), & i \leq i_0; \\ m_i q_i, & i \geq i_0. \end{cases} \quad \square$$

Wegener’s results are based on the fact that random OBDDs are essentially worst-case shaped whenever n is such that for all levels i , either $k_i = o(m_i)$ or $k_i = \omega(m_i)$ holds. But the situation in Main Theorem 1 (ii) is different. We have to be especially careful about those n for which an i exists such that k_i and m_i are of roughly the same size, because then $q_i \neq o(1)$ and $q_i \neq 1 + o(1)$. These levels have the form $i_\delta \in \mathbb{N}$ for some sufficiently small $|\delta|$. (See Definition 3.3 for a formal statement.)

These so-called *critical levels* are what we will be mainly concerned with in this section. More precisely, we say that a level j is critical if the size of $E(X_j)$ is bigger than a constant fraction of W . The existence of such a critical level is what makes the strong Shannon effect break down occasionally.

There seems to be no way to compute $E(X_j)/W$ directly. But using the results from Section 2, we will look at the ratio $E(X_j)/2^L$ instead. Of course, for $|\delta| \rightarrow +\infty$, no level i_δ can be critical, since then $w_j = o(W)$. But the ‘‘threshold’’ for δ which we will determine is much smaller. We will show that there is always at most *one* critical level.

This analysis culminates in Theorem 3.8, which then leads to a first qualitative result about the expected qOBDD size with respect to a fixed variable ordering of random Boolean functions, Theorem 3.13.

Large deviations from the expected value of Y will be considered in Section 4. The effect of the deletion rule will be analysed in Section 5, and finally the proof of the Main Theorem 1 on OBDDs with optimal variable orderings will be completed in Section 5.4.

Many asymptotics derived in this section will be used throughout the whole dissertation. For the convenience of the reader, we have tabulated some of them in the appendix, see page 116.

Before we delve into the technicalities, we explain the idea of the proof in an easy special case.

Proposition 3.2. For $n = 2^h + h$ and $h \nearrow +\infty$, $\mathbb{E}(X)/W = \Omega(1)$, so the strong Shannon effect does not hold. More precisely,

$$\liminf_{\substack{n=2^h+h \\ h \nearrow +\infty}} \frac{\mathbb{E}(X_{i_0})}{2^L} \geq \frac{1}{e}, \quad \text{and} \quad \liminf_{\substack{n=2^h+h \\ h \nearrow +\infty}} \frac{\mathbb{E}(X)}{W} \geq \frac{1}{2e}.$$

Proof. Recall that $L(2^h + h) = 2^h$, which implies that the *critical point* $i_0 = 2^h + 1 \in \mathbb{N}$ is also a *level*. Observe that $w_{i_0} = k_{i_0} = m_{i_0} = 2^L = 2^{2^h}$. By Proposition 3.1, we have

$$\mathbb{E}(X_{i_0}) = m_{i_0} \left(1 - \frac{1}{m_{i_0}}\right)^{k_{i_0}} = 2^{2^h} \left(1 - \frac{1}{2^{2^h}}\right)^{2^{2^h}} \sim \frac{2^{2^h}}{e},$$

and by Theorem 2.4, $W(2^h + h) \sim 2 \cdot 2^{2^h}$. □

The tightness of these bounds will be shown in Theorem 3.8. Some prerequisites are developed in the next section.

3.1 Prerequisites

The proof of Proposition 3.2 was easy because $L(2^h + h) = 2^h$ is an integer (and therefore i_0 , too). This is no longer true for arbitrary n . We will show that taking more than a constant number of steps away from the “bad” values $n = 2^h + h$ is enough to guarantee that the strong Shannon effect holds, and that any constant number does not suffice.

Stated in another way, the difficulty that arises in the proof for general n is that the *critical point* i_0 does no longer coincide with the *critical level* of the qOBDD, which in fact can be $\lfloor i_0 \rfloor$ or $\lceil i_0 \rceil$, depending on n . Therefore, we introduce a parameter $\delta'(n) \in \mathbb{R}$ such that $i_{\delta'(n)}$ will (hopefully) be the critical level, if there is any.

Definition 3.3. Let $\delta'(n)$ denote the gap between $L(n)$ and the next natural number, i. e.

$$\delta'(n) := \ell - L(n),$$

where $\ell \in \mathbb{N}$ is the unique element of $\left] L(n) - \frac{1}{2}, L(n) + \frac{1}{2} \right] \cap \mathbb{N}$, and write

$$i' := i_{\delta'}.$$

By definition, $i_{\delta'(n)} = i_0 + \delta'(n) = \ell + 1$ is the integer nearest to i_0 . In case of a tie, we round up, so $\delta'(n) \in \left] \frac{-1}{2}, \frac{1}{2} \right]$.

Observe that $\delta'(2^h + h) = 0$, and this was the easy case we considered in Proposition 3.2. We will see that as $|\delta'|$ gets larger, $\mathbf{E}(X_{i_{\delta'}})$ becomes negligible compared to 2^L . Another fact remaining to be shown is that there is always at most one critical level. Since we do not want to state theorems in terms of δ' , we have to find out how large δ' is, depending on n .

In the proofs of the bounds on the worst-case size in Section 2, we wrote n in the form $n = 2^h + h + a$ or $n = b2^h + h$ for appropriately chosen h , a , and b . We will continue using this idea and introduce two other parameters $h(n)$ and $a(n)$ such that $n = 2^{h(n)} + h(n) + a(n)$. As one might have expected, it turns out that a is closely related to δ' .

There is one difficulty with this approach, however. As n grows from $2^{h'} + h'$ to $2^{h'+1} + h' + 1$, the parameter $\delta'(n)$ first goes up from 0 to about $1/2$, then suddenly jumps down to about $-1/2$ and finally becomes 0 again (remember that $L(n)$ grows slightly slower than n). To make things easier, we want to ensure that the jumps of $a(n)$ are at the same positions as those of $\delta'(n)$. It turns out that the precise position of the jumps is not important. Therefore, in the following definition we simply require that a and δ' have the same sign.

Definition 3.4. For $n \in \mathbb{N}$, we define $h(n) \in \mathbb{N}$ and $a(n) \in \mathbb{Z}$ by the requirements that $n = 2^{h(n)} + h(n) + a(n)$ and $a(n) \cdot \delta'(n) \geq 0$, and $|a(n)|$ be minimal under the first two conditions.

We note some immediate consequences of Definitions 3.3 and 3.4.

Proposition 3.5.

- (i) The positions where the parameter $\delta'(n)$ jumps are of the form $n \sim \sqrt{2}2^{h'}$, $h' \in \mathbb{N}$.
- (ii) Assume that we are given two sequences $(h'_t | t \in \mathbb{N})$ and $(a'_t | t \in \mathbb{N})$ such that $a'_t = o(2^{h'_t})$ as $t \rightarrow +\infty$, and let $n_t := 2^{h'_t} + h'_t + a'_t$. Then $h(n_t) = h'_t$ and $a(n_t) = a'_t$.
- (iii) For large n , we have $|a(n)| \leq 0.42 \cdot 2^{h(n)}$ and thus, $n = \Theta(2^{h(n)})$.

Proof. Assertion (i): Writing $L(n) = n - \log L(n)$ using (2.1), we see that $|\delta'(n)| = 1/2 + o(1) \pmod{1}$ if and only if $\log L(n) = 1/2 + o(1) \pmod{1}$. Let h' be the largest integer such that $2^{h'} < L(n)$. Then $\log L(n) = \log \frac{L(n)}{2^{h'}} \pmod{1}$, and the claim follows because $1/2 = \log \sqrt{2}$.

Assertion (ii): Immediate from (i).

Assertion (iii): Let h' be as in the proof of (i). If $\delta'(n) \sim 1/2$, then $h(n) = h'$ and $a(n) \sim (\sqrt{2} - 1)2^{h(n)}$. If $\delta'(n) \sim -1/2$, then $h(n) = h' + 1$ and $a(n) \sim (1/\sqrt{2} - 1)2^{h(n)}$. So

$$\limsup_{n \in \mathbb{N}} \frac{|a(n)|}{2^{h(n)}} = \max \left\{ \sqrt{2} - 1, 1 - \frac{1}{\sqrt{2}} \right\} < 0.42. \quad \square$$

How are the two parameters δ' and a related? Given an a , we would like to know how big δ' is, at least in an approximate sense. It turns out that a and δ' are approximately proportional as long as we do not move away “too far” from the “nice” values $n = 2^h + h$. This connection is made explicit by the definition of $\tilde{a}(n)$ and the lemma that follows it. The notation $\tilde{a}(n)$ was chosen to emphasise that $\tilde{a}(n)$ is of about the same size as $a(n)$. The ‘magic’ scaling factor is just $\log'(1)$.

Definition 3.6. We define

$$\delta_x := \frac{x \log e}{L(n)}$$

and

$$\tilde{a}(n) := \frac{\delta'(n) L(n)}{\log e}$$

such that $\delta' = \delta_{\tilde{a}}$ is satisfied.

By the following lemma, “not too far” means “ $o(n)$ ”. Since we also want to prove when the strong Shannon effect *does* hold, we must show that in the other case, δ' is not small enough. It suffices to prove $|\tilde{a}| \rightarrow +\infty$.

Lemma 3.7.

- (i) If $a = o(n)$, then $\tilde{a} = a + O(a^2/n) \sim a$.
- (ii) If $a \rightarrow \pm\infty$, then $\tilde{a} \rightarrow \pm\infty$.

Proof. Assertion (i): Let $h = h(n)$ and $L = L(n)$. To determine $\delta'(n)$, we expand L using the functional equation (2.1):

$$L = n - \log L = \underbrace{n - h}_{\in \mathbb{N}} - \log \frac{L}{2^h}. \quad (3.1)$$

We claim that $\delta'(n) = \frac{\log L}{2^h} = o(1)$. Using (2.1) once more, we find that

$$\log \frac{L}{2^h} = \log \frac{2^h + h + a - \log L}{2^h} = \log \left(1 + \frac{h - \log L + a}{2^h} \right). \quad (3.2)$$

Since

$$\log(1 + x) = x \log e + O(x^2) \quad (3.3)$$

for $x = o(1)$, and $L(n) \sim n$,

$$h - \log L = h - \log(2^h(1 + o(1))) = -\log(1 + o(1)) = o(1).$$

So

$$\log \frac{L}{2^h} = \log \left(1 + \frac{a + o(1)}{2^h} \right) = \frac{a \log e}{2^h} \left(1 + O\left(\frac{a}{2^h}\right) \right) = o(1)$$

is indeed the fractional part of L as was suggested in (3.1), and $\delta'(n) = a \log e / L + O(a^2/L^2) \sim a \log e / L$. Therefore, $\tilde{a} = L \delta' / \log e = a + O(a^2/n) \sim a$.

Assertion (ii): To prove the contrapositive, assume that there exists an infinite subsequence of n for which $\tilde{a} = O(1)$. This means that we have

$$L = \ell - \delta' = \ell - \delta_{\tilde{a}}, \quad (3.4)$$

where $\ell = \ell(n) \in \mathbb{N}$ and $\tilde{a} = \tilde{a}(n) = O(1)$. For notational convenience, assume that the subsequence is equal to the original one. Application of L^{-1} to both sides of (3.4) gives

$$n = \ell - \delta_{\tilde{a}} + \log(\ell - \delta_{\tilde{a}}). \quad (3.5)$$

Observe that $|\delta_{\tilde{a}}|$ is rather small; we have

$$\delta_{\tilde{a}} = O(\tilde{a})/L = O(1/n).$$

So we can rewrite (3.5) as

$$n = \ell + \log \ell - c,$$

with a small correction term c , whose size is only

$$c := \delta_{\tilde{a}} + \log \frac{\ell}{\ell - \delta_{\tilde{a}}} = O(1/n) \quad (3.6)$$

by (3.3). Therefore, we can guess that $h(n)$ is equal to

$$h' := \log \ell - c = n - \ell \in \mathbb{N}. \quad (3.7)$$

So far, we know that h' is an integer close to $\log \ell$. But what about $2^{h'}$ and ℓ ? Expressing ℓ in terms of c and h' , we find

$$\ell = 2^{\log \ell} = 2^{h'+c} = 2^{h'+O(1/n)} = 2^{h'}(1 + O(1/n)) = 2^{h'} + O(2^{h'/n}),$$

and using (3.4) and (3.6), we have

$$2^{h'} = 2^{\log \ell - c} = 2^{\log(L+O(1)) - O(1/n)} = (L + O(1))2^{O(1/n)} \sim L = O(n).$$

Therefore,

$$a' := n - 2^{h'} - h' = n - \ell - h' + O(1) = O(1).$$

But this implies that $h(n) = h'$ and $a(n) = a' = O(1)$ by Proposition 3.5 (ii). Going from the subsequence back to the original sequence, we have shown that $|\tilde{a}| \not\rightarrow +\infty$ implies $|a| \not\rightarrow +\infty$. By definition, a and \tilde{a} have the same sign. \square

Actually, the following considerations suggest (but not prove) that Lemma 3.7 (ii) can be improved to say that $\tilde{a}(n)/a(n) = \Theta(1)$ as $n \rightarrow +\infty$. We take up the estimations from the proof of Proposition 3.5. If $\delta'(n) \sim 1/2$, then $\tilde{a}(n) \sim \frac{L(n)}{2 \log e} \sim \frac{\sqrt{2}}{2 \log e} 2^{h(n)}$. If $\delta'(n) \sim -1/2$, then $\tilde{a}(n) \sim \frac{-L(n)}{2 \log e} \sim \frac{-\sqrt{2}}{4 \log e} 2^{h(n)}$. So if we take for granted that these are the extremal parametrisations, we can conclude that \tilde{a}/a lies between $\frac{\sqrt{2}}{2 \log e} / (\sqrt{2} - 1) + o(1) < 1.19$ and $\frac{-\sqrt{2}}{4 \log e} / (\frac{1}{\sqrt{2}} - 1) + o(1) > 0.83$.

3.2 The Expected Size of the Levels of a Random qOBDD

Now we are prepared to extend the idea of Proposition 3.2 to the case of general n . Assertions (i) and (ii) of Theorem 3.8 are concerned with the size of the (possibly) critical level i' , while Assertion (iii) says that the other levels altogether contribute only

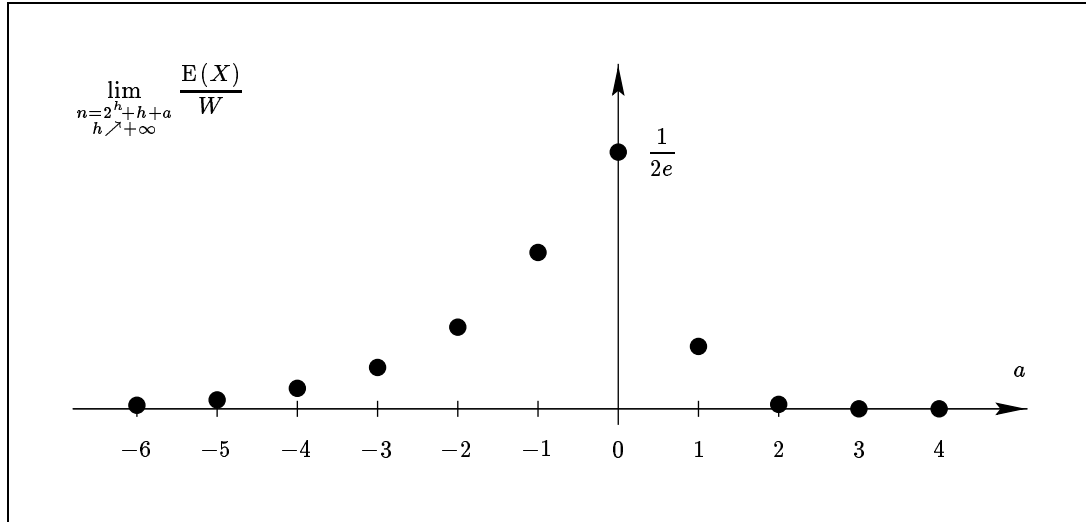


Fig. 9. When the strong Shannon effect does not hold (Corollary 3.11).

$o(1)$ to $E(X)/2^L$. Note that (i) implies the existence of a limit for constant a , which is not at all obvious. See Fig. 9 for an illustration.

Theorem 3.8.

(i) For sequences of n such that $a(n) = o(\sqrt{n})$,

$$\frac{E(X_{i'})}{2^L} = \begin{cases} 2^{-a+o(1)} (e^{-2^{a+o(1)}} - 1) + 1, & a \leq 0; \\ 2^{-a+o(1)} e^{-2^{a+o(1)}}, & a \geq 0. \end{cases}$$

(ii) For sequences of n such that $|a(n)| \rightarrow +\infty$, $\frac{E(X_{i'})}{2^L} = o(1)$.

(iii) For all sequences of n , $\frac{E(X - X_{i'})}{2^L} = 2^{-\Omega(\sqrt{n})}$.

Proof. Let $k := k_{i'}$ and $m := m_{i'}$. Since we want to use Proposition 3.1, we need upper and lower bounds on $q := q_{i'}$.

Estimating q :

Writing

$$q = \left(1 - \frac{1}{m}\right)^k = \left(1 - \frac{1}{m}\right)^{m \frac{k}{m}} = \left(1 - \frac{1}{m}\right)^{(m-1) \frac{k}{(m-1)}}$$

and using the inequalities $(1 - 1/x)^{x-1} > 1/e > (1 - 1/x)^x$, valid for $x \geq 2$, we see that

$$e^{-k/m} > q > e^{-k/(m-1)} \geq e^{-k/m} e^{-2k/m^2} \geq e^{-k/m} \left(1 - \frac{2k}{m^2}\right). \quad (3.8)$$

Therefore,

$$q = e^{-k/m} \left(1 - \left|O\left(\frac{k}{m^2}\right)\right|\right), \quad (3.9)$$

which is $e^{-k/m} (1 - |O(1/m)|)$, if $a \leq 0$ (because then $k \leq m$).

Proof of Assertion (i):

In view of the proof of Assertions (ii) and (iii), the estimations we derive to prove Assertion (i) will be based on the weaker assumption $\tilde{a} = o(n)$ and not require that $a = o(\sqrt{n})$. By Lemma 3.7 (i), $\tilde{a} = a + o(1)$ for $a = o(\sqrt{n})$. We will investigate positive and negative values of \tilde{a} separately.

Estimating $k/2^L$ and $m/2^L$ for $\tilde{a} = o(n)$:

Clearly,

$$\frac{k}{2^L} = 2^{\delta'} = e^{\tilde{a}/L} = e^{o(1)} \sim 1, \quad (3.10)$$

because $n \sim L(n)$.—To estimate $m/2^L$, we apply the Taylor approximation $e^x = 1 + x + O(x^2)$ for $x \rightarrow 0$, which yields

$$(1 - 2^{-\delta'})L = (1 - e^{-\tilde{a}/L})L = \tilde{a} + r \sim \tilde{a} \quad (3.11)$$

for some

$$r = r(n) = O(\tilde{a}^2/L) = O(\tilde{a}^2/n) = o(\tilde{a}). \quad (3.12)$$

So

$$\frac{m}{2^L} = 2^{(2^{-\delta'}-1)L} = 2^{-\tilde{a}-r}. \quad (3.13)$$

The case $0 \geq \tilde{a} = o(n)$:

In this case, $E(X_{i_{\delta'}}) = k - m(1 - q)$ by Proposition 3.1, and the asymptotic (3.9) for q implies that

$$\begin{aligned} E(X_{i_{\delta'}}) &= k - m(1 - q) = k - m(1 - e^{-k/m}(1 - O(k/m^2))) \\ &= k - m(1 - e^{-k/m}) + O(k/m). \end{aligned} \quad (3.14)$$

By substituting (3.10) and (3.13) into (3.14), we obtain

$$\frac{\mathbf{E}(X_{i'})}{2^L} = 2^{-\tilde{a}-r} \left(e^{-2^{\tilde{a}+r} e^{\tilde{a}/L}} - 1 \right) + e^{\tilde{a}/L} + O(2^{-L}). \quad (3.15)$$

In particular, (i) follows for $a = -|o(\sqrt{n})|$.

The case $0 \leq \tilde{a} = o(n)$:

In this case, $\mathbf{E}(X_{i'}) = mq$ by Proposition 3.1 and $q = e^{-k/m}(1 + O(k/m^2))$ by (3.9).

Since $\delta' = \delta_{\tilde{a}} = o(n) \log e/L = o(1)$, we have

$$\frac{k}{m^2} = \frac{2^{\delta'+L}}{(2^{2^{-\delta'}L})^2} \leq 2^{o(1)+L-2 \cdot 2^{o(1)}L} = 2^{-(1+o(1))L} = o(1),$$

and the asymptotic for q simplifies to $q \sim e^{-k/m}$. Together with (3.10) and (3.13), we obtain

$$\frac{\mathbf{E}(X_{i'})}{2^L} = \frac{mq}{2^L} \sim \frac{m}{2^L} e^{-k/m} = \frac{2^{-\tilde{a}-r}}{e^{2^{\tilde{a}+r+o(1)}}}, \quad (3.16)$$

where $r = O(\tilde{a}^2/L)$ by (3.12). In particular, (i) follows for $a = |o(\sqrt{n})|$.

Proof of Assertion (ii), first part:

We split the sequence of n into four subsequences, depending on whether $|\tilde{a}(n)| \leq \sqrt{n}$ or $|\tilde{a}(n)| > \sqrt{n}$ and whether $\tilde{a}(n) \geq 0$ or $\tilde{a}(n) \leq 0$. Lemma 3.7 (ii) tells us that $|\tilde{a}(n)| \rightarrow +\infty$, because $|a(n)| \rightarrow +\infty$.

For the two subsequences satisfying $|a(n)| \leq \sqrt{n}$, the estimations from the proof of Assertion (i) can be applied, since we only used the premise $\tilde{a} = o(n)$ in the proof. Also, we still have $a \sim \tilde{a}$, so (3.12) implies $r = O(1)$.

The case $0 \leq \tilde{a}(n) \rightarrow +\infty \wedge \tilde{a}(n) \leq \sqrt{n}$:

Since $\tilde{a} + r \rightarrow +\infty$, $\mathbf{E}(X_{i'})/2^L \rightarrow 0$ follows immediately from (3.16).

The case $0 \geq \tilde{a}(n) \rightarrow -\infty \wedge \tilde{a}(n) \geq -\sqrt{n}$:

In this case, $\tilde{a} + r \rightarrow -\infty$, so $2^{\tilde{a}+r} = o(1)$. Using (3.15) and the Taylor approximation $e^x = 1 + x + (1 + o(1))x^2/2$ for $x \rightarrow 0$ we see that

$$\begin{aligned} \frac{\mathbf{E}(X_{i'})}{2^L} &= 2^{-\tilde{a}-r} \left(-2^{\tilde{a}+r} e^{\tilde{a}/L} + 2^{2(\tilde{a}+r)-1} e^{2\tilde{a}/L} (1 + o(1)) \right) + e^{\tilde{a}/L} + O(2^{-L}) \\ &= 2^{\tilde{a}+r-1} e^{2\tilde{a}/L} (1 + o(1)) + O(2^{-L}) = o(1). \end{aligned} \quad (3.17)$$

So far we have proved Assertion (ii) for $|\tilde{a}| \leq \sqrt{n}$. To complete the proof of Theorem 3.8 (ii) and (iii), we need a lemma. ...

Lemma 3.9. Assume that $n \rightarrow +\infty$ and $j = j(n) \in [n]$ is a level such that

$$|j - i_0| \geq \frac{\sqrt{n} \log e}{L(n)},$$

where i_0 denotes the critical point. Let $a' := (j - i_0)L(n)/\log e$. Then

$$\frac{\mathbb{E}(X_j)}{2^L} \leq \frac{\mathbb{E}(X_j)}{w_j} = \begin{cases} 2^{a'+O(1)} = 2^{-\Omega(\sqrt{n})}, & j < i_0; \\ e^{-2^{\Omega(\sqrt{n})}}, & j > i_0. \end{cases}$$

Proof. Recall that the definition of $\tilde{a}(n)$ was made such that $i_{\delta'} = i_{\tilde{a}}$. In this lemma, we are no longer concerned with $i_{\delta'}$, but an arbitrary level j . Nevertheless, we can define

$$a' := \frac{(j - i_0) L(n)}{\log e} \quad (3.18)$$

such that $j = i_{\delta_{a'}}$ is satisfied. Then $|j - i_0| \geq \sqrt{n} \log e / L(n)$ is equivalent to $|a'| \geq \sqrt{n}$. We write $k := k_j$ and $m := m_j$ and $q := q_j$ and $L := L(n)$. We consider two cases: $a' \leq -\sqrt{n}$ and $a' \geq \sqrt{n}$. Recall that w_j is given by (2.6) and bounded by 2^L .

First, assume that $a' \leq -\sqrt{n}$. Then

$$\frac{k}{m} \leq \frac{2^L}{m} = 2^{(1-2^{-\delta_{a'}})L} = 2^{(1-e^{-a'/L})L} \leq 2^{a'} = 2^{-\Omega(\sqrt{n})}. \quad (3.19)$$

In particular, we have

$$q = e^{-k/m} (1 + O(k/m^2)) = e^{-k/m} (1 + O(2^{a'}/m))$$

by (3.9). Therefore, analogously to (3.14), it holds

$$\begin{aligned} \mathbb{E}(X_j) &= k - m(1 - q) = k - m(1 - e^{-k/m}(1 - O(2^{a'}/m))) \\ &= k - m(1 - e^{-k/m}) + O(2^{a'}). \end{aligned}$$

Using $e^x = 1 + x + (1 + o(1))x^2/2$ for $x \rightarrow 0$, we get

$$\begin{aligned} \mathbb{E}(X_j) &= k - m \left(1 - \left(1 - \frac{k}{m} + \frac{k^2}{2m^2} (1 + o(1)) \right) \right) + O(2^{a'}) \\ &= \frac{k^2}{2m} (1 + o(1)) + O(2^{a'}). \end{aligned} \quad (3.20)$$

Here the leading term is bounded by

$$\frac{k^2}{2m} = k \cdot \frac{k}{2m} \leq k \cdot 2^{a'-1}$$

because of (3.19). It follows that

$$\frac{\mathbb{E}(X_j)}{w_j} = \frac{\mathbb{E}(X_j)}{k} \leq 2^{a'+O(1)} = 2^{-\Omega(\sqrt{n})} \quad (3.21)$$

as claimed.

Now for the second case, $a' \geq \sqrt{n}$. By (3.8), and since $k \geq 2^L$,

$$\mathbb{E}(X_j) = mq \leq me^{-k/m} \leq me^{-2^L/m}. \quad (3.22)$$

Here,

$$\frac{2^L}{m} = 2^{(1-2^{-\delta_{a'}})L}$$

and using $e^{-x} \leq \frac{1}{1+x}$, we get

$$(1 - 2^{-\delta_{a'}})L = (1 - e^{-a'/L})L \geq \frac{a'/L}{1 + a'/L}L \geq \frac{\sqrt{n}}{1 + \sqrt{n}/L} = \Omega(\sqrt{n}).$$

Therefore,

$$\frac{\mathbb{E}(X_j)}{w_j} = \frac{\mathbb{E}(X_j)}{m} \leq e^{-2^L/m} \leq 2^{-2^{\Omega(\sqrt{n})}}.$$

□ (Lemma 3.9)

Proof of Theorem 3.8, continued:

Proof of Assertion (ii), conclusion:

The remaining subsequences satisfy $|\tilde{a}(n)| > \sqrt{n}$ and are therefore covered by Lemma 3.9—just set $j := i'$.

Proof of Assertion (iii):

If $j \in [n] \setminus \{i_{\delta'}\}$, then $|j - i_0| \geq 1/2$ holds by definition of δ' . So by Lemma 3.9,

$$\frac{\mathbb{E}(X - X_{i'})}{2^L} = \sum_{j \in [n] \setminus \{i'\}} \frac{\mathbb{E}(X_j)}{2^L} = n \cdot 2^{-\Omega(\sqrt{n})} = 2^{-\Omega(\sqrt{n})}.$$

□ (Theorem 3.8)

Theorem 3.8 gives us a fairly complete overview of the ‘expected’ shape of a random qOBDD. Above and below $i_{\delta'}$, the levels are essentially full. If δ' is sufficiently small, i. e., $\delta' = O(1/n)$, then the expected size of the critical level $i_{\delta'}$ is by a factor $1 - \Omega(1)$ smaller than its worst-case width. For later reference, we summarise these observations in a corollary.

Corollary 3.10. Assume that $n \rightarrow +\infty$ and $j = j(n) \in [n]$.

(i) For all j ,

$$1 - \frac{1}{e} \leq \frac{\mathbf{E}(Y_j)}{w_j} \leq 1.$$

(ii) If $|j - i_0| = \omega(1/n)$, then

$$\frac{\mathbf{E}(Y_j)}{w_j} \sim 1.$$

Proof. *Assertion (i):* It follows from Theorem 3.8 (in particular, equations (3.15) and (3.16) from the proof of its Assertion (i)) that $\mathbf{E}(X_j)/w_j$ is maximised at $j = i_0$, where $\mathbf{E}(X_{i_0}) = w_{i_0}/e$. Thus,

$$\frac{\mathbf{E}(Y_j)}{w_j} = \frac{w_j - \mathbf{E}(X_j)}{w_j} \geq 1 - \frac{1}{e}.$$

Assertion (ii): This is immediate from Theorem 3.8 (ii) and Lemma 3.9. \square

If $a(n) = o(\sqrt{n})$, then we can determine the ratio of the expected qOBDD size to its worst-case size very precisely.

Corollary 3.11. If $n \rightarrow +\infty$ is such that $a(n) = o(\sqrt{n})$, then

$$\frac{\mathbf{E}(Y)}{W} = \begin{cases} 1 - \frac{2^{-a+o(1)} (e^{-2^{a+o(1)}} - 1) + 1}{2 + o(1)}, & a \leq 0; \\ 1 - \frac{2^{-a+o(1)} e^{-2^{a+o(1)}}}{1 + 2^{-a} + o(1)}, & a \geq 0. \end{cases}$$

Proof. Direct plug-in from Theorems 2.4 and 3.8. \square

The decrease rate of $\mathbf{E}(X)/W$ (shown in Fig. 9 on page 47) is doubly exponential for $a \rightarrow +\infty$, but only exponential for $a \rightarrow -\infty$ by the next proposition. While this makes little difference for *constant* a and the results on OBDDs, it will become important for FBDDs (Theorem 10.9).

Proposition 3.12. If $\alpha \rightarrow -\infty$, then $2^{-\alpha} (e^{-2^\alpha} - 1) + 1 \sim 2^{\alpha-1}$.

Proof. Using $e^x = 1 + x + \frac{x^2}{2}(1 + o(1))$ for $x \rightarrow 0$,

$$2^{-\alpha}(e^{-2^\alpha} - 1) + 1 = 2^{-\alpha}(-2^\alpha + 2^{2\alpha-1}(1 + o(1))) + 1 = 2^{\alpha-1}(1 + o(1)). \quad \square$$

3.3 The Expected Size of Random qOBDDs with a Fixed Variable Ordering

We extract a qualitative result from the preceding quantitative analysis of the expected qOBDD size with a fixed variable ordering of a random Boolean Function. Using Theorem 3.8, we obtain a yes-or-no answer to the question: “For which n is the expected size of the qOBDD with a fixed variable ordering for a random Boolean function equal to the worst-case size up to terms of lower order?”

Theorem 3.13. Let $B := \bigcup_{h \in \mathbb{N}} [2^h + h - d(h) .. 2^h + h + d(h)]$ and $A := \mathbb{N} \setminus B$.

(i) If $n \rightarrow +\infty$ such that $n \in A$ for some $d(h) \rightarrow +\infty$, then

$$\frac{\mathbf{E}(Y)}{W} = 1 - o(1).$$

(ii) If $n \rightarrow +\infty$ such that $n \in B$ for some $d(h) = O(1)$, then

$$\frac{\mathbf{E}(Y)}{W} = 1 - \Omega(1).$$

Proof. We have $\mathbf{E}(X)/W = \Theta(\mathbf{E}(X)/2^L)$, since by Theorem 2.6, $W = \Theta(2^L)$. By Theorem 3.8 (iii), all levels except i' are negligible (i. e., they contribute only $o(1)$ to $\mathbf{E}(X)/2^L$).

Assertion (i): Since $d(h) \rightarrow +\infty$ and n is a sequence chosen from the set A , we have $|a(n)| \rightarrow +\infty$ as $n \rightarrow +\infty$. Therefore, $\lim_n \mathbf{E}(X_{i'})/2^L = 0$ by Theorem 3.8 (ii), and we are done.

Assertion (ii): Since $d(h) = O(1)$ and n is a sequence chosen from the set B , we have $a(n) = O(1)$. By partitioning the sequence of n into subsequences, we may assume that $a(n)$ is a constant. The subsequences may have finite or infinite length, but only a finite number of subsequences can be infinitely long, because the original sequence satisfied $a(n) = O(1)$. For each infinite subsequence of integers n where $a(n)$ is a constant, we have proved in Theorem 3.8 (i) that $\lim_n \mathbf{E}(X_{i'})/2^L > 0$. So the original sequence satisfies $\liminf_n \mathbf{E}(X_{i'})/2^L > 0$, i. e., $\mathbf{E}(X_{i'})/2^L = \Omega(1)$. \square

Knowing the expected size is nice, but in order to determine for which n the strong Shannon effect holds, we must also understand the probability of large deviations from the expected size. This will be investigated in the next sections.

4 The Strong Shannon Effect for qOBDDs with Optimal Variable Orderings

To find out for which n the strong Shannon effect holds for (reduced) OBDDs with optimal variable orderings, we must show that for *almost all* Boolean functions, *all* variable orderings lead to an OBDD size which is equal to the worst-case size W' up to a factor of $1 + o(1)$. Remember that $W' \sim W$. We have already seen that the *expected* qOBDD size for a *fixed* variable ordering is $\sim W$ if and only if $|a(n)| \rightarrow +\infty$ (Theorem 3.13). The next step is to consider qOBDDs with *optimal* variable orderings. Our approach is to prove that for a random Boolean function, with high probability *all* variable orderings lead to *almost the same* qOBDD size. (Here we apply Azuma's inequality.) In particular, an optimal variable ordering does only a little better than the canonical one. The effect of the deletion rule still will not be considered. This is postponed to Section 5 for technical reasons.

4.1 Azuma's Inequality

Azuma's martingale inequality is an important tool in random graph theory and the analysis of random discrete structures. It enables us to prove strong concentration results for random variables which satisfy a Lipschitz condition. Also, the probability space must 'factorise' in an appropriate way. Here we give a combinatorial formulation that totally avoids the language of martingale theory.

Theorem 4.1. (Azuma's Inequality, see e. g. [AS91, Theorem. 7.4.2])

If B is a finite domain and $S: B^k \rightarrow \mathbb{R}$ is a function satisfying the 'Lipschitz condition'

$$\forall b, b' \in B^k : \#\{j \mid b_j \neq b'_j\} \leq 1 \rightarrow |S(b) - S(b')| \leq 1 \quad (4.1)$$

and the coordinates of b are chosen independently at random, then

$$\Pr_b \left(|S(b) - \mathbb{E}(S)| \geq \lambda \sqrt{k} \right) \leq 2 e^{-\lambda^2/2}. \quad \square$$

The application of Azuma's inequality to the urn experiment is straightforward. This is not a new idea; a similar result was already given in [AS91].

Corollary 4.2. Consider an urn experiment where k balls are thrown independently uniformly at random into m urns, and denote by y the number of non-empty urns. Then

$$\Pr \left(|y - \mathbf{E}(y)| \geq \lambda \sqrt{k} \right) \leq 2e^{-\lambda^2/2}.$$

Proof. Denote a random assignment of balls to urns by $b: [k] \rightarrow [m]$ and let $y(b) := \#b[k]$ be the number of non-empty urns for this particular assignment. Clearly, y satisfies the Lipschitz condition (4.1), since the number of non-empty urns can only change by 1 if we move a ball from one urn to another. Therefore, Theorem 4.1 is applicable. \square

It is a bit surprising that the bound we obtained via Azuma's inequality does not involve any dependency on m at all.

4.2 Large Deviations from the Expected qOBDD size

Using Theorem 4.2, we can prove a very strong concentration result for the expected size of the quasireduced OBDD (respecting the canonical variable ordering) of a random Boolean function. The next theorem asserts that the probability that Y is somewhat more than $\sqrt{\mathbf{E}(Y)}$ apart from its expected value is only doubly exponentially small (in n).

Theorem 4.3. For every $c > 0$,

$$\Pr \left(|Y - \mathbf{E}(Y)| \geq n2^{\frac{1+c}{2}L} \right) \leq 2ne^{-2^{cL/4}}.$$

Proof. At each level j , we have an urn experiment where k_j balls are thrown into m_j urns, and Y_j is the number of nodes at level j of the qOBDD as well as the number of non-empty urns. So by Corollary 4.2,

$$\Pr \left(|Y_j - \mathbf{E}(Y_j)| \geq \lambda \sqrt{k_j} \right) \leq 2e^{-\lambda^2/2}. \quad (4.2)$$

For best results, we consider two cases. If $j > i_1$, then $Y_j \leq m_1 = 2^{L/2}$. If $j \leq i_1$, then $k_j \leq 2^{L+1}$, and using (4.2) with $\lambda := 2^{\frac{1+c}{2}L}/\sqrt{k_j}$, we get

$$\Pr \left(|Y_j - \mathbf{E}(Y_j)| \geq 2^{\frac{1+c}{2}L} \right) \leq 2e^{-2^{cL/4}}, \quad (4.3)$$

since $\lambda^2/2 \geq 2^{(1+c)L-L-1}/2 = 2^{cL}/4$. So

$$\Pr\left(\exists j \in [n] : |Y_j - \mathbf{E}(Y_j)| \geq 2^{\frac{1+c}{2}L}\right) \leq n \cdot 2e^{-2^{cL}/4}, \quad (4.4)$$

and the theorem follows. \square

It is easy to see that using Azuma's inequality, one cannot improve the cut-off point beyond $\omega(\sqrt{k_j})$, and $k_j = \Omega(2^L)$ for $j = i'$. The question arises whether a weaker (not doubly exponential) probability bound is provable for some cut-off point $O(2^{(\frac{1}{2}-\Omega(1))L})$. The answer is "at least in general: no", because $Y_{i'}$ is asymptotically normally distributed for certain parametrisations of n . For example, if $n = 2^h + h$, then $i' = 2^h + 1$ and $k_{i'} = m_{i'} = 2^h$, and the distribution of Y_{2^h+1} is asymptotically normal by Theorem I.3.1 of [KSC78]. Note however, that this result can only be applied if the ratio k/m is a *constant*, because it makes no assertion on the convergency rate.

For another kind of large deviation inequalities, see Section 6.2.

4.3 Optimal qOBDDs for Random Boolean Functions

Define $Y_*(f) := \min_{\pi} Y_{\pi}(f)$, where the index π runs over all variable orderings. $Y_*(f)$ is the minimal size of a qOBDD representing f . Clearly, $\mathbf{E}(Y_{\pi})$ does not depend on π . The next theorem asserts that for most Boolean functions, choosing an optimal variable ordering gives little improvement. The probability that Y_* is somewhat more than $\sqrt{\mathbf{E}(Y)}$ apart from $\mathbf{E}(Y)$ is doubly exponentially small (in n). Of course, $\mathbf{E}(Y_*) \leq \mathbf{E}(Y)$. We consider two-sided deviations for simplicity.

Theorem 4.4. For every $c > 0$,

$$\Pr\left(|Y_* - \mathbf{E}(Y)| \geq n2^{\frac{1+c}{2}L}\right) \leq e^{-2^{cL}/4 + O(n \log n)}.$$

Proof. If $|Y_* - \mathbf{E}(Y)|$ is large, then there exists a variable ordering π such that $|Y_{\pi} - \mathbf{E}(Y)|$ is large. For each variable ordering, there is only a doubly exponentially small fraction of exceptional Boolean functions. So we simply multiply the probability bound from Theorem 4.3 by $n! < n^n = e^{n \log n}$. \square

From a larger perspective, the important facts are that $2^{\frac{1+c}{2}L} = o(W)$ and $W \cdot e^{-2^{cL}/4 + O(n \log n)} = o(1)$. As a consequence, Theorem 3.13 carries over (without any changes) to the case of optimal variable orderings. This answers the question: "For which n is the *expected optimal* qOBDD size of a random Boolean function equal to the worst-case size up to terms of lower order?" But Theorem 4.4 is much stronger. We get

a yes-or-no answer to the question: “For which n does the *strong Shannon effect* hold for qOBDDs with optimal variable orderings?” This is an assertion about Y_* , not $\mathbb{E}(Y_*)$.

Theorem 4.5. Let $n \rightarrow +\infty$. Then

$$\Pr(Y_* = (1 - o(1))W) \sim 1 \leftrightarrow |a(n)| = \omega(1).$$

That is, the strong Shannon effect holds for qOBDDs with optimal variable orderings if and only if n is such that $|a(n)| \rightarrow +\infty$. \square

In the next section, we will take the effect of the deletion rule into account.

5 The Effect of the Deletion Rule

So far, we know that the minimal qOBDD size is $\sim W$ if and only if $|a(n)| \rightarrow +\infty$ (Theorem 4.5). The analysis of the influence of the deletion rule has been postponed until now for two reasons: Firstly, the deletion rule is only applicable after some nodes have been merged (namely, the two successors of the node which is about to be deleted). The second reason is of a more technical kind: We need to know the size of the levels in order to estimate the expected number of deletions.

The general message of this section is that for most Boolean functions, the deletion rule gives only a comparatively small amount of reduction. The proofs are organised very similar to the analysis of the merging rule. First, we consider random reduced OBDDs with a fixed variable ordering and compute $E(X'_j)$, the expected number of deletions, for each level j . Then we prove a large deviation inequality for X'_j , using a result of Chvátal [Chv79]. Again, we obtain a doubly exponential probability bound, which then leads to the proof of the Main Theorem 1 for *reduced* OBDDs with optimal variable orderings.

The result that the deletion rule does not give much reduction when applied to a qOBDD is not new [Weg94], but here we give a much more refined analysis and better probability bounds.

5.1 The Expected Reduction by the Deletion Rule

The expected reduction by the deletion rule is described level wise in the following theorem.

Theorem 5.1. Assume that $n \rightarrow +\infty$ and $j \in [n]$. Let $\delta := j - i_0$ and $a' := \delta L(n) / \log e$, so that $j = i_{\delta a'}$.

$$(i) \text{ We have } E(X'_j) \sim c \cdot \begin{cases} 2^{\delta + (1 - 2^{-\delta - 1})L}, & \delta \leq 0; \\ 2^{2^{-\delta - 1}L}, & \delta \geq 0. \end{cases}$$

$$(ii) \text{ If } a' = o(n), \text{ then } E(X'_j) \sim c \cdot \sqrt{2^{L - |a'|}}.$$

(iii) In both (i) and (ii), we have

$$c < 1 \quad \text{and} \quad c \sim \frac{E(Y_j) + O(2^{\frac{1+o(1)}{2}L})}{w_j},$$

where the size of $E(Y_j)/w_j$ is given by Corollary 3.10.

Proof. The number of nodes deleted at level i can be computed by an (other) urn experiment. Among the m_i subfunctions which are possible at level i of the qOBDD there are

m_{i+1} functions that do not depend essentially on the variable x_i . Since the merging rule has already been applied, we have a situation in which Y_j balls are chosen *without replacement* from an urn containing $m_j - m_{j+1}$ “white” and m_{j+1} “black” balls. The black balls correspond to the nodes which are deleted at level j . Therefore, X'_j is hypergeometrically distributed with parameters

$$p_j := \frac{m_{j+1}}{m_j} = 2^{(2^{-\delta-1}-2^{-\delta})L} = 2^{-2^{-\delta-1}L}$$

and Y_j . Note, however, that Y_j is itself a random variable. So far, we can only conclude that

$$\forall y \in [0 .. w_j]: \mathbf{E}(X'_j | Y_j = y) = p_j y$$

by the formula for the expected value of the hypergeometric distribution. To get around this difficulty, we apply Theorem 4.3 with, say, $c := 1/\log L$. Actually, we did not only prove that $|Y - \mathbf{E}(Y)|$ is small with high probability, but that each $|Y_j - \mathbf{E}(Y_j)|$ is small with high probability (see (4.4)). It follows that

$$|Y_j - \mathbf{E}(Y_j)| \leq 2^{\frac{1+1/\log L}{2}L} = \sqrt{2^{(1+o(1))L}} \quad (5.1)$$

with probability

$$1 - 2e^{-2^{L/\log L}/4} = 1 - o(1/w_j).$$

Even if (5.1) does not hold, we still have the upper bound $Y_j \leq w_j$. Therefore,

$$\mathbf{E}(X'_j) = p_j \left(\mathbf{E}(Y_j) + O(\sqrt{2^{(1+o(1))L}}) \right)$$

and of course, $\mathbf{E}(X'_j) \leq p_j w_j$. This gives us the size of c which was claimed in (iii). The rest of the proof is just straightforward calculation.

For $\delta \leq 0$, we have

$$p_j w_j = p_j k_j = 2^{-2^{-\delta-1}L+L+\delta}. \quad (5.2)$$

For $\delta \geq 0$, we have

$$p_j w_j = p_j m_j = m_{j+1} = 2^{2^{-\delta-1}L}. \quad (5.3)$$

This proves Assertion (i).

Now assume that $a' = o(L)$. Then for $a' \leq 0$,

$$p_j w_j = 2^{((1-2^{-\delta a'})^{L+L})/2+\delta a'} = \sqrt{2^{L+a'+o(1)}}$$

by (5.2) and (3.11). For $a' \geq 0$,

$$p_j w_j = 2^{((2^{-\delta a'}-1)^{L+L})/2} = \sqrt{2^{L-a'+o(1)}}$$

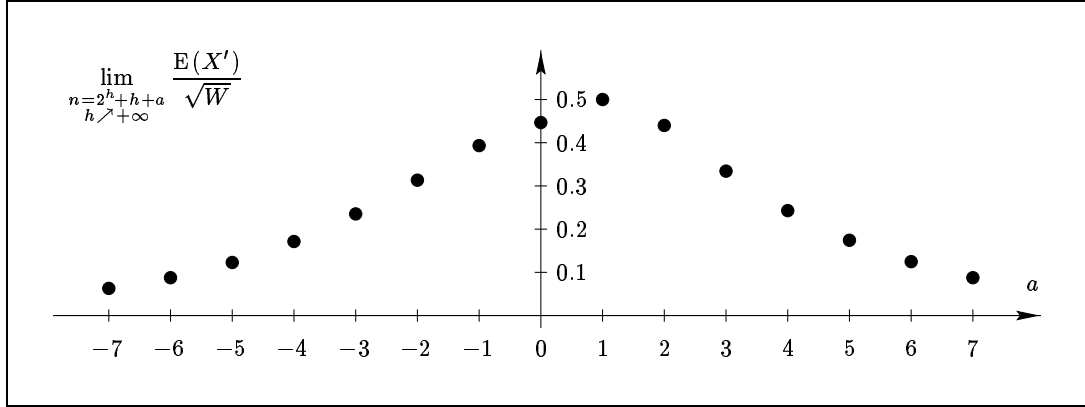


Fig. 10. The effect of the deletion rule for n close to $2^h + h$ (Corollary 5.2).

by (5.3) and (3.11). This proves Assertion (ii). \square

Of course, we are especially interested in the situation near $n = 2^{h'} + h'$. For $a(n) = o(n)$, we have $E(X'_j) = 2^{\frac{1+o(1)}{2}L}$ by Theorem 5.1 (ii), whereas $W = \Theta(2^L)$ by Theorem 2.4. We can even compute the size of the ratio $E(X'_j)/\sqrt{W}$ fairly exactly. See Fig. 10.

Corollary 5.2. If $n \rightarrow +\infty$ is such that $a(n) = o(n)$, then

$$\frac{E(X')}{\sqrt{W}} \sim \left(1 - e^{-2^{a+o(1)}}\right) \cdot \begin{cases} 1/\sqrt{2^{a+1}}, & a \leq 0; \\ 1/\sqrt{2^a + 1}, & a \geq 0. \end{cases}$$

Proof. We begin by showing that $E(X') \sim E(X'_{i'})$. To prove this, observe that if $j \leq i_{-1/2}$, then

$$E(X'_j) = O(2^{(1-2^{-1/2})L}) = o(2^{0.4L/n}),$$

and if $j \geq i_{1/2}$, then

$$E(X'_j) = O(2^{2^{-3/2}L}) = o(2^{0.4L/n}).$$

Therefore,

$$E\left(\sum_{j \in [n] \setminus \{i'\}} X'_j\right) = o(2^{0.4L}),$$

which is $o(E(X'_{i'}))$, because

$$E(X'_{i'}) \sim \frac{E(Y_{i'})}{w_{i'}} \sqrt{2^{L-|a|}} = \Omega(2^{\frac{1+o(1)}{2}L}). \quad (5.4)$$

The precise size of $\mathbb{E}(Y_{i'})/w_{i'}$ was computed in the proof of Theorem 3.8 (ii). By (3.10), (3.12), (3.13), (3.15), and (3.16), we have

$$\frac{\mathbb{E}(Y_{i'})}{w_{i'}} \sim \begin{cases} 2^{-a+o(1)} \left(1 - e^{-2^{a+o(1)}}\right), & a \leq 0; \\ 1 - e^{-2^{a+o(1)}}, & a \geq 0. \end{cases} \quad (5.5)$$

Now we combine (5.4) and (5.5) with the asymptotics for $W/2^L$ given in Theorem 2.4. □

Oddly, $\mathbb{E}(X')/\sqrt{W}$ is maximised not at $a = 0$, but at $a = 1$ (with value $\doteq 0.499$). The reason for this is that $\mathbb{E}(X_{i'})/2^L$ declines *doubly* exponentially as $a \rightarrow +\infty$, so the increase of $\mathbb{E}(Y_{i'})/w_{i'}$ temporarily compensates the decay of $\sqrt{2^{L-|a|}}$. (See also Fig. 9 on page 47.)

The size of $\mathbb{E}(X')$ for various parametrisations of n will be investigated in Section 6, where it is also compared with $\mathbb{E}(X)$. It turns out that $\mathbb{E}(X')$ is always concentrated on the two levels $\lfloor i_0 \rfloor$ and $\lceil i_0 \rceil$. Also, the amount of reduction achieved by the deletion rule changes in a periodic way, as was the case for the merging rule. However, the shape of the oscillations is quite different.

5.2 Large Deviations from the Expected Reduction by the Deletion Rule

To estimate the probability of large deviations from the expected amount of reduction by the deletion rule, we apply a large deviation inequality for hypergeometrically distributed random variables due to Chvátal. The statement of the following theorem is already adapted to our setting.

Theorem 5.3. (Chvátal [Chv79]) Assume that y balls are chosen without replacement from an urn containing a fraction of p black balls, and denote by x' the number of black balls chosen. Then x' is a hypergeometrically distributed random variable with parameters p and y , mean $\mathbb{E}(x') = py$, and for all $\varepsilon \geq 0$, we have

$$\Pr(x' \geq (p + \varepsilon)y) \leq e^{-2\varepsilon^2 y}. \quad \square$$

The next theorem gives a doubly exponential bound on the probability that X' is somewhat bigger than the square root of W or $\mathbb{E}(Y)$, which are both $\Theta(2^{n/2})$.

Theorem 5.4. For every $c > 0$,

$$\Pr \left(X' \geq n2^{\frac{1+c}{2}L} \right) \leq e^{-(2+o(1))2^{cL}}.$$

Proof. Recall that X'_j is hypergeometrically distributed with parameters $p_j = \frac{m_{j+1}}{m_j}$ and Y_j . Using Theorem 5.3 and $Y_j \leq w_j$, we can estimate the situation at each level by

$$\begin{aligned} \Pr \left(X'_j \geq (p_j + \varepsilon_j)w_j \right) &= \sum_{y=0}^{w_j} \underbrace{\Pr \left(X'_j \geq (p_j + \varepsilon_j)w_j \mid Y_j = y \right)}_{\leq \Pr \left(X'_j \geq (p_j + \varepsilon_j)w_j \mid Y_j = w_j \right)} \Pr \left(Y_j = y \right) \\ &\leq \Pr \left(X'_j \geq (p_j + \varepsilon_j)w_j \mid Y_j = w_j \right) \\ &\leq e^{-2\varepsilon_j^2 w_j}. \end{aligned} \quad (5.6)$$

A sufficient condition for $X' \leq n2^{\frac{1+c}{2}L}$ is that the inequality

$$X'_j \leq 2^{\frac{1+c}{2}L} \quad (5.7)$$

is satisfied for all j . Let $\delta := j - i_0$ (so that $j = i_\delta$). If $j > i_1$, then (5.7) holds trivially, because

$$X'_j \leq w_j = m_j = 2^{2^{-\delta}L} \leq 2^{\frac{1}{2}L}.$$

For $j \leq i_1$, we set

$$\varepsilon_j := \frac{2^{\frac{1+c}{2}L}}{w_j} - p_j,$$

so that

$$(p_j + \varepsilon_j)w_j = 2^{\frac{1+c}{2}L}.$$

We claim that $\varepsilon_j \geq 0$. Observe that for $j \geq i_0$, we have

$$p_j = 2^{-2^{-\delta-1}L} = \sqrt{2^{-2^{-\delta}L}} = \frac{1}{\sqrt{w_j}}. \quad (5.8)$$

Comparing the logarithms, we find that

$$\log \frac{2^{\frac{1+c}{2}L}}{w_j p_j} = \log \left(2^{\frac{1+c}{2}L} p_j \right) = \left(\frac{1+c}{2} - 2^{-\delta-1} \right) L \geq \frac{c}{2} L > 0. \quad (5.9)$$

So $\varepsilon_j \geq 0$ for $j \geq i_0$. For $j \leq i_0$, we use the facts that $\varepsilon_{i_0} \geq 0$ and that for every n , the mappings $j \mapsto k_j$ and $j \mapsto p_j$ are isotone. Therefore, $\varepsilon_j \geq \varepsilon_{i_0} \geq 0$.

So (5.6) is applicable, and we have

$$\Pr \left(X'_j \geq 2^{\frac{1+c}{2}L} \right) \leq e^{-2\varepsilon_j^2 w_j} \quad (5.10)$$

for all j .

To lower bound $\varepsilon_j^2 w_j$, we again consider two cases. If $i_0 \leq j \leq i_1$, then by (5.8),

$$\varepsilon_j^2 w_j = \left(\frac{2^{\frac{1+c}{2}L}}{w_j} - p_j \right)^2 w_j = 2^{(1+c)L} p_j^2 - 2^{\frac{1+c}{2}L+1} p_j + 1 = t_j^2 - 2t_j + 1,$$

where $t_j := 2^{\frac{1+c}{2}L} p_j \geq 2^{\frac{c}{2}L}$ by (5.9). Therefore,

$$\varepsilon_j^2 w_j \geq 2^{cL} (1 + o(1)).$$

If $j \leq i_0$, then

$$\begin{aligned} \varepsilon_j^2 w_j &= \left(\frac{2^{\frac{1+c}{2}L}}{w_j} - p_j \right)^2 w_j \\ &= \left(\sqrt{\frac{2^{(1+c)L}}{w_j}} + O(1) \right)^2 \geq \left(\sqrt{\frac{2^{(1+c)L}}{2^L}} + O(1) \right)^2 \sim 2^{cL}. \end{aligned}$$

So we have shown that $\varepsilon_j^2 w_j \geq 2^{cL} (1 + o(1))$ for $j \leq i_1$. Using (5.10), the probability that (5.7) fails for at least one level is bounded by

$$n \cdot e^{-2 \cdot 2^{cL} (1+o(1))} = e^{-(2+o(1))2^{cL}}, \quad (5.11)$$

and the theorem follows. \square

The attentive reader might wonder why the proof of Theorem 5.4 makes no reference to Theorem 5.1, but uses only the trivial inequality $\mathbb{E}(Y_j) \leq w_j$ instead (see (5.6)). The reason is that it would make no difference for the main result we are aiming for, the sharp concentration of the reduced OBDD size. We already saw that the cut-off point cannot be improved beyond $2^{L/2}$ when analysing large deviations from the expected quasireduced OBDD size. (See the remarks following Theorem 4.3.) — Theorem 5.1 will be used in Section 6 and the proof of Theorem 9.6.

Another kind of large deviation bounds will be derived in Section 6.2.

5.3 Optimal Variable Orderings and the Deletion Rule

Using the results from Section 5.2, we now show that the gap between Y and Z is comparatively small for *all* variable orderings, with high probability. This follows easily from

the doubly exponential bound on the probability of large deviations, $e^{-(2+o(1))2^{cL}}$ in Theorem 5.4.

We define $X'_*(f)$ to be the maximal number of nodes that can be deleted from $\text{qOBDD}_\pi(f)$, for any variable ordering π . So formally, $X'_*(f) := \max_\pi X'_\pi(f)$.

Theorem 5.5. For every $c > 0$,

$$\Pr \left(X'_* \geq n2^{\frac{1+c}{2}L} \right) \leq e^{-(2+o(1))2^{cL}}.$$

Proof. All we have to do is to multiply the probability bound from the fixed variable ordering case by the number of variable orders, which is $n! < n^n = e^{n \ln(n)}$. We get a probability bound of

$$e^{n \ln n} \cdot e^{-(2+o(1))2^{cL}} = e^{-(2+o(1))2^{cL}}, \quad (5.11')$$

which proves the corollary. \square

We make no attempt to optimise the factor $2 + o(1)$ in the exponent or even the double exponent cL in the probability bound in Theorems 5.4 and 5.5 any further, because it has no consequences for our main results.

The deletion rule will be revisited in the context of FBDDs in Section 8.

5.4 The Weak Shannon Effect for OBDDs

Recall that the weak Shannon effect asserts that almost all Boolean functions have almost the same size for a certain kind of representation. If we combine the large deviation results Theorem 4.4 and Theorem 5.5, we obtain the following corollary.

Corollary 5.6. Let $Z_*(f)$ denote the minimal OBDD size of a Boolean function f . Then

$$\Pr \left(|Z_* - \mathbb{E}(Y)| \geq 2n2^{\frac{1+c}{2}L} \right) \leq e^{-2^{cL}/4 + O(n \log n)},$$

and since $\mathbb{E}(Y) = \Omega(2^L)$, the *weak Shannon effect* holds for OBDDs (and qOBDDs) with optimal variable orderings representing random Boolean functions. \square

Finally, the Main Theorem 1 on OBDDs with optimal variable orderings follows from Theorem 3.13 and Corollary 5.6.

6 Comparing the Reduction Rules

While Main Theorem 1 is sufficient to decide whether the strong Shannon effect holds for a particular sequence of n (or not), a closer inspection shows that $n = 2^h + h$ is not the only interesting point in the evolution of random OBDDs. Two other (more subtle) phase transitions take place as n passes from $2^h + 2h - 1$ to $2^h + 2h$ and at $n = (\frac{5}{4} + o(1))2^h$. These results are complementary to Main Theorem 1 in the sense that they make assertions about where the ‘Shannon gap’ $X + X'$ is *minimised*.

6.1 Comparing the Expected Amount of Reduction

Once we know that $E(X)$ is ‘big’ near $2^h + h$ and ‘small’ (compared with W) otherwise, it is natural to ask: “How small can $E(X)$ become?” The following theorem tells us that $E(X) = \Theta(n^2)$ for $n = 2^h + 2h$, which is *very* small compared with the worst-case size $\Theta(2^n/n)$. Another remarkable fact is that $E(X) \geq 2^{0.278n}$ for the parametrisation $n = 2^h + 2h - 1$, which has only one level less. This rapid phase transition is due to the merging rule solely; note that for both parameterisations, $E(X') = 2^{(1/2+o(1))n}$ by Theorem 5.1 (i), since $|\delta| \geq 1$ implies $E(X'_{i_\delta}) = O(2^{L/4})$. In other words, there is a phase transition for *quasireduced* OBDDs, but not for *reduced* OBDDs. The following theorem looks at parametrisations of n ‘around’ $2^h + 2h$. See Fig. 11 for an illustration.

Theorem 6.1.

(i) Assume that $n = 2^{h(n)} + c(n)h(n) \rightarrow +\infty$, where $2 \leq c(n) = o(n)$. Then

$$E(X) = n^{\tilde{c}(n)},$$

where $\tilde{c}(n) \sim 2(c(n) - 1)$.

(ii) Assume that $n = 2^{h(n)} + 2h(n) - c'(n) \rightarrow +\infty$, where $c'(n) \geq 1$ and $0 < h(n) - c'(n) \rightarrow +\infty$. Then

$$E(X) = 2^{\tilde{c}'n},$$

where $\tilde{c}'(n) = 1 - 2^{-c'(n)} \log e + o(1)$.

Proof. The assumptions in (i) and (ii) ensure that $a(n) = o(n)$, so Lemma 3.7 (i) is applicable and we have $\delta'(n) = \delta_{\tilde{a}}$ with $\tilde{a} \sim a$. Also, $0 < a \rightarrow +\infty$, because $a = (c - 1)h \geq h \rightarrow +\infty$ in (i) and $0 < a = h - c' \rightarrow +\infty$ in (ii). Note that $\delta' \in [0, \frac{1}{2}]$. It turns out that the proportions of $E(X_{i'})$ and $E(X_{i'-1})$ become inverted as a passes from $h - 1$ to h .

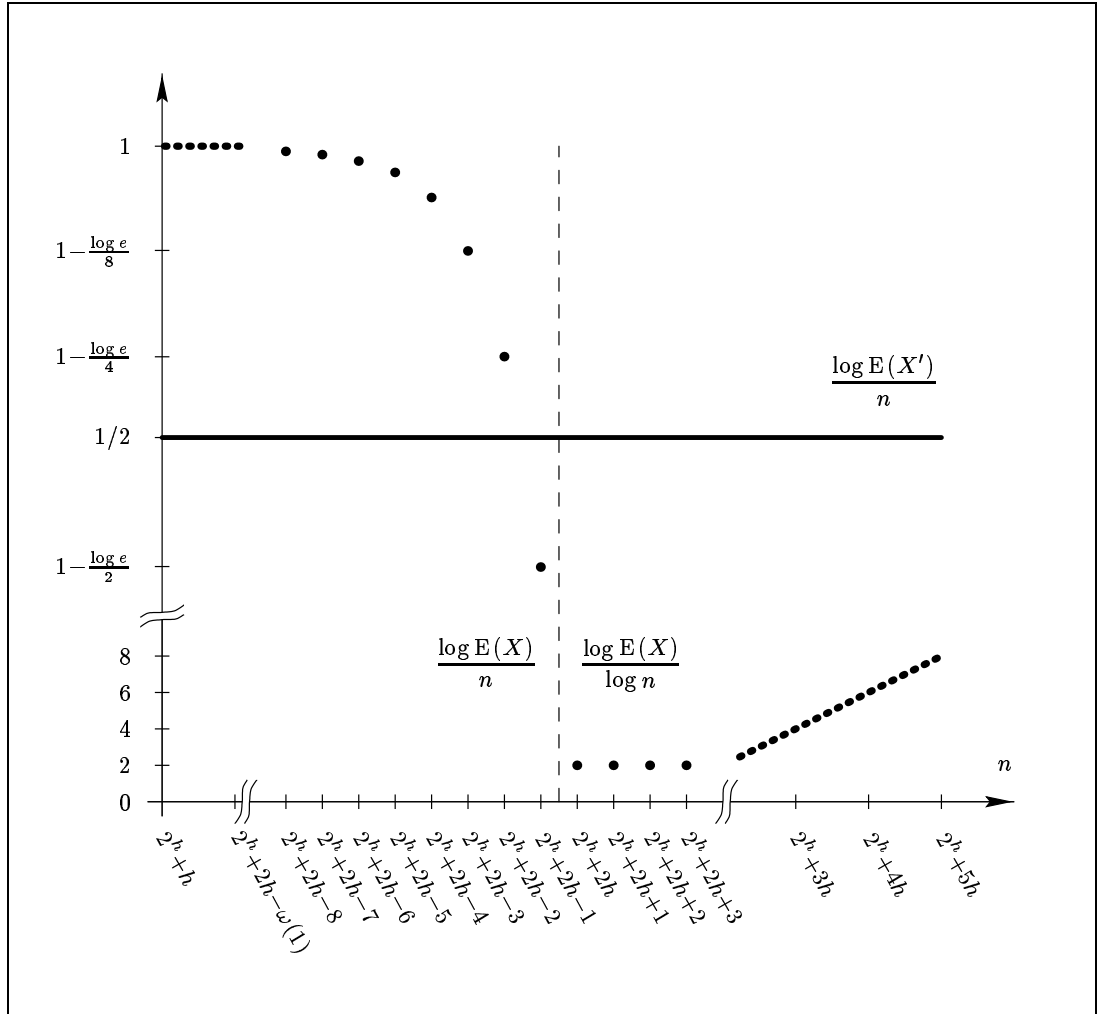


Fig. 11. Comparison of $E(X)$ and $E(X')$ around $n = 2^h + 2h$ (Theorem 6.1).
The dashed line marks the phase transition.

For level i' , we apply the estimates from the proof of Theorem 3.8 (i), which are valid under the assumption $\tilde{a} = o(n)$. By (3.16) and (3.12), we have

$$E(X_{i'}) \sim 2^{L-\tilde{a}+O(\tilde{a}^2/n)} e^{-2\tilde{a}+O(\tilde{a}^2/n)+o(1)}. \quad (6.1)$$

For level $i' - 1$, we apply the estimates from the proof of Lemma 3.9. If we plug in the special value $j = i_{\delta'-1}$ into (3.18), then

$$a' = \frac{(\delta'(n) - 1)L(n)}{\log e} = \tilde{a}(n) - \frac{L(n)}{\log e}, \quad (6.2)$$

which implies $a' \leq -\sqrt{n}$. So the leading term in (3.20) is

$$\frac{k_{i'-1}^2}{2m_{i'-1}} = \left(2^{(\delta'-1)+L}\right)^2 2^{-2^{1-\delta'}L-1} = 2^{2\delta_{\tilde{a}}-3+2(1-2^{-\delta_{\tilde{a}}})L} = 2^{2\tilde{a}-3+O(\tilde{a}^2/n)}$$

by (3.11), and

$$\mathbb{E}(X_{i'-1}) \sim 2^{2\tilde{a}-3+O(\tilde{a}^2/n)}. \quad (6.3)$$

Now we know $\mathbb{E}(X_{i'})$ and $\mathbb{E}(X_{i'-1})$, but we also have to show that all the other levels $j \in [n] \setminus \{i'-1, i'\} \subseteq [n] \setminus [i_{-3/2}, i_1]$ are negligible.

If $j \geq i_1$, then $\mathbb{E}(X_j) \leq me^{-2^{L/m}}$ by (3.22). But $m_j \leq 2^{L/2}$, so $\mathbb{E}(X_j) \leq 2^{L/2}e^{-2^{L/2}} = o(1/n)$.

If $j \leq i_{-3/2}$, then $\mathbb{E}(X_j) \leq 2^{L-a'+O(1)}$ by (3.21). Similar to (6.2), we find that

$$a' \leq \frac{-3}{2} \frac{L(n)}{\log e} \leq -1.01 L(n),$$

so $\mathbb{E}(X_j) \leq 2^{-L/100+O(1)} = o(1/n)$. (Note that $\log e \doteq 1.44$.)

Altogether, we have shown that

$$\sum_{j \in [n] \setminus \{i'-1, i'\}} \mathbb{E}(X_j) \leq n \cdot o(1/n) = o(1).$$

Assertion (i): Using (6.3) and the facts that $\tilde{a} \sim a \rightarrow +\infty$, $a = o(n)$, $a = (c-1)h$, and $h \sim \log n$, we find that

$$\begin{aligned} \mathbb{E}(X_{i'-1}) &\sim 2^{2\tilde{a}-3+O(\tilde{a}^2/n)} = 2^{2a(1+o(1))} \\ &= 2^{2(c-1)h(1+o(1))} = 2^{2(c-1)\log n(1+o(1))} = n^{2(c-1)(1+o(1))}, \end{aligned}$$

which has already the right exponent of n . We claim that $\mathbb{E}(X_{i'}) = o(1)$. By (6.1), we have

$$\mathbb{E}(X_{i'}) \leq 2^{L - \log e 2^{\tilde{a}+O(\tilde{a}^2/n)+o(1)}}.$$

We consider two cases. If $a > 2h$ and n is large enough, then

$$\tilde{a} + O(\tilde{a}^2/n) + o(1) = \tilde{a}(1 + o(1)) + o(1) \geq h.$$

If $h \leq a \leq 2h$, then $\tilde{a}^2/n = o(1)$, so

$$\tilde{a} + O(\tilde{a}^2/n) + o(1) \geq h + o(1).$$

In both cases, we have

$$L - \log e 2^{\tilde{a}+O(\tilde{a}^2/n)+o(1)} \leq L - \log e 2^{h+o(1)} \rightarrow -\infty,$$

since $L \sim 2^h$. So indeed, $\mathbb{E}(X_{i'}) = o(1)$, and Assertion (i) follows.

Assertion (ii): Using (6.3) and the facts that $0 < a = h - c < h$ and $\tilde{a} \sim a = o(\sqrt{n})$, we see that $\mathbb{E}(X_{i'-1}) = O(n^2)$. To estimate $\mathbb{E}(X_{i'})$, observe that $\tilde{a} = a + o(1)$ by Lemma 3.7 (i). So (6.1) implies that

$$\begin{aligned} \mathbb{E}(X_{i'}) &\sim 2^{L-\tilde{a}+o(1)} e^{2^{\tilde{a}+o(1)}} = 2^{L-h+c+o(1)} e^{2^{h-c+o(1)}} \\ &= 2^{L-h+c+o(1)-\log e^{2^{h-c+o(1)}}} = 2^{(1-\log e^{2^{-c+o(1)}})L}, \end{aligned}$$

which is $\Omega(2^{(1-\log e^{2^{-c+o(1)}})L})$, because $c \geq 1$. Assertion (ii) follows. \square

We have seen that both $\mathbb{E}(X) \gg \mathbb{E}(X')$ and $\mathbb{E}(X) \ll \mathbb{E}(X')$ are possible. So for finding the point where the ‘‘Shannon gap’’ is minimised, we must take the deletion rule into account as well. The proportion of $\mathbb{E}(X)$ and $\mathbb{E}(X')$ for small $a \geq 0$ is already clear from Theorems 6.1 and 5.1. In Theorem 6.2, we look at the values of n which are not covered by Theorem 6.1. This is illustrated by Fig. 12.

Theorem 6.2. Assume that $n \rightarrow +\infty$ and $h' \in \mathbb{N}$ and b are such that $n = b2^{h'} + h'$ and $b \in [1 + h'/2^{h'}, 2]$. (So n is in the range $[2^{h'} + 2h' .. 2^{h'+1} + h']$.)

Then

- (i) $\frac{\log \mathbb{E}(X)}{n} = 2 - \frac{2}{b} + o(1)$, and
- (ii) $\frac{\log \mathbb{E}(X')}{n} = \max \left\{ \frac{1}{2b}, 1 - \frac{1}{b} \right\} + o(1)$.

Proof. First, we introduce a new parameter $\delta'' = \delta''(n)$, which will be used in this proof only. Since $L(n) \sim n \sim b2^{h'}$, we have

$$L(n) = n - \log L(n) = n - h - \log b + o(1). \quad (6.4)$$

Here $n - h \in \mathbb{N}$, so $\delta'(n) = \log b + o(1) \pmod{1}$. The important levels are $i_{\delta''}$ and $i_{\delta''-1}$, where

$$\delta'' := n - h - L = \log b + o(1)$$

by (6.4). Note that $\delta'' \in \{\delta', \delta' + 1\}$. By working with δ'' instead of δ' , we avoid difficulties with the jumps of δ' , see Lemma 3.5. Using the monotonicity of L , one can easily show that δ'' jumps from ~ 1 to 0 as n passes from $2^{h'} + h' - 1$ to $2^{h'} + h'$ and that $\delta'' \in [0, 1]$.

Assertion (i): If $h \leq a < n^{2/3}$, then $b \sim 1$, and $\log \mathbb{E}(X)/n = o(1)$ follows since $\mathbb{E}(X) = n^{O(n^{2/3})}$ by Theorem 6.1 (i). So we can assume without loss of generality that

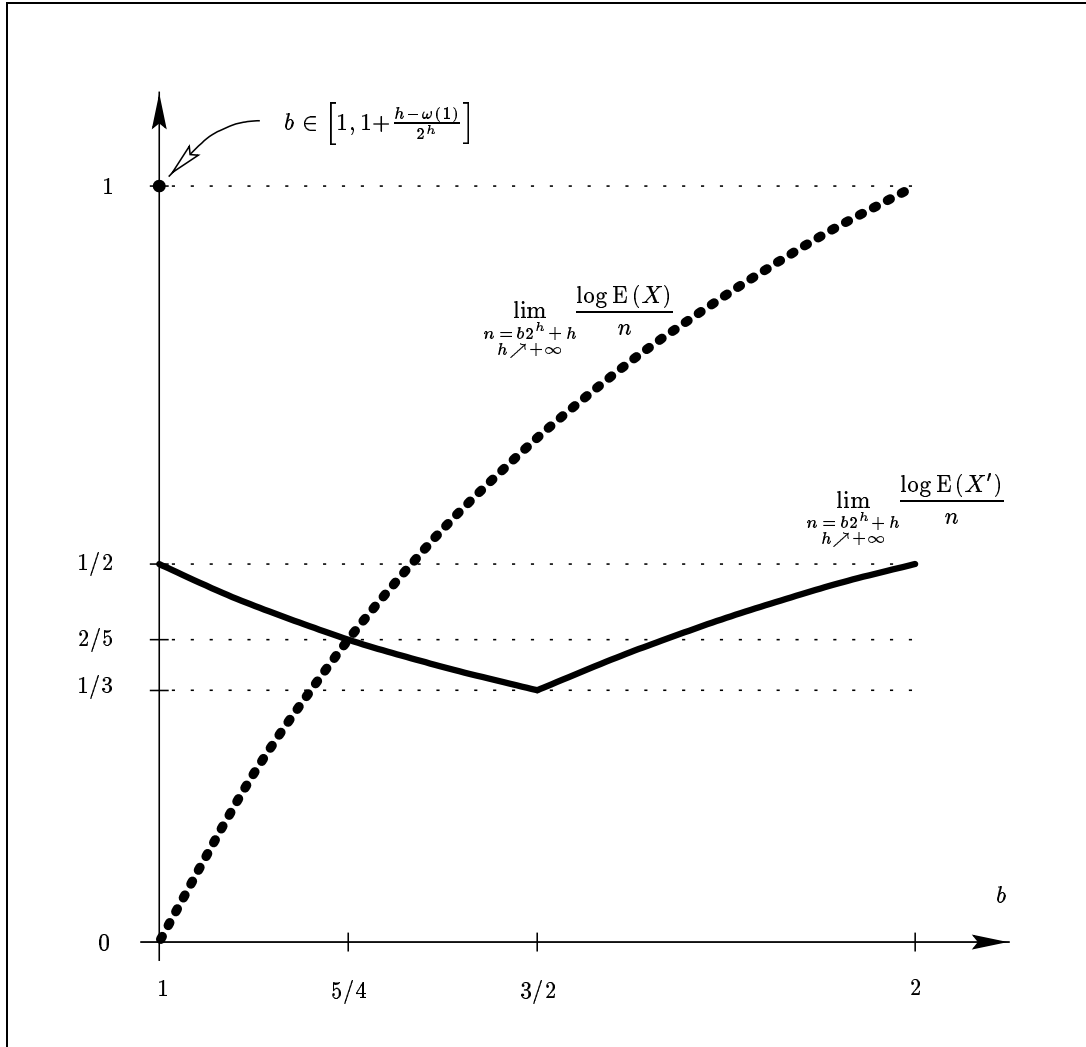


Fig. 12. A global view on $E(X)$ and $E(X')$ (Theorem 6.2). The dot at $(1, 1)$ refers to Theorem 6.1.

$\delta'' \geq \sqrt{n} \log e/L$. Lemma 3.9 implies that $E(X_j) \leq 2^n e^{-2^{\Omega(\sqrt{n})}} = o(1)$ for all levels $j \geq i_{\delta\sqrt{n}}$, which includes $i_{\delta''}$.

We claim that

$$E(X_{i_\delta}) = 2^{(2-2^{-\delta}+o(1))L} + O(1) \quad (6.5)$$

if $\delta \leq 0$.—Choose a' such that $\delta = \delta_{a'}$. For $0 \geq a' \geq -\sqrt{n}$ we can apply the asymptotic (3.17), and get

$$E(X_{i_\delta}) = 2^{L-a'(1+o(1))} e^{o(1)} + O(1) = 2^{n(1+o(1))} + O(1)$$

as claimed in (6.5). If $a' \leq -\sqrt{n}$, then by (3.20) from the proof of Lemma 3.9, we have

$$E(X_{i_\delta}) = \frac{k_{i_\delta}^2}{2m_{i_\delta}}(1 + o(1)) + o(1),$$

and again,

$$\mathbb{E}(X_{i_\delta}) = 2^{2(\delta+L)-1-2^{-\delta}L} + o(1) = 2^{(2-2^{-\delta}+o(1))n} + o(1)$$

as claimed in (6.5).

By (6.5), $\mathbb{E}(X_j) = 2^{o(L)}$ for all levels $j \leq i_{-1}$. Therefore, $\mathbb{E}(X) = \mathbb{E}(X_{i_{\delta''-1}}) + 2^{o(L)}$. Since $\delta'' - 1 = \log b - 1 + o(1)$, we have

$$\mathbb{E}(X_{i_{\delta''-1}}) = 2^{(2-2^{1-\log b}+o(1))n} = 2^{(2-2/b+o(1))n},$$

and Assertion (i) follows.

Assertion (ii): Theorem 5.1 (i) implies that $\mathbb{E}(X'_{i_\delta}) = O(2^{L/4})$ if $|\delta| \geq 1$. Also, we have

$$\mathbb{E}(X'_{i_{\delta''}}) = \Theta\left(2^{2^{-\log b-1+o(1)}L}\right) = \Theta\left(2^{(1/2b+o(1))L}\right)$$

and

$$\mathbb{E}(X'_{i_{\delta''-1}}) = \Theta\left(2^{\log b-1+(1-2^{-\log b}+o(1))L}\right) = \Theta\left(2^{(1-1/b+o(1))L}\right).$$

Therefore,

$$\mathbb{E}(X') = \Theta\left(2^{(1/2b+o(1))L} + 2^{(1-1/b+o(1))L}\right) + O(n2^{L/4}).$$

Simple computations show that

$$\min_{b \in [1,2]} \max \left\{ \frac{1}{2b}, 1 - \frac{1}{b} \right\} = \frac{1}{3},$$

and Assertion (ii) follows. □

The results of this section are summarised in Corollaries 6.3 and 6.4.

Corollary 6.3. The minimising parametrisations for the ‘Shannon gap’ are

- (i) for $\mathbb{E}(X)$: $n = 2^h + 2h$, where $\mathbb{E}(X) = n^{2+o(1)}$;
- (ii) for $\mathbb{E}(X')$: $n = \left(\frac{3}{2} + o(1)\right) 2^h$, where $\mathbb{E}(X') = 2^{(\frac{1}{3}+o(1))n}$;
- (iii) for $\mathbb{E}(X) + \mathbb{E}(X')$: $n = \left(\frac{5}{4} + o(1)\right) 2^h$, where $\mathbb{E}(X) + \mathbb{E}(X') = 2^{(\frac{2}{5}+o(1))n}$. □

Corollary 6.4. Assume that $n \rightarrow +\infty$ and write $n = 2^{h(n)} + h(n) + a(n) = b2^{h'} + h'$, where $h' \in \mathbb{N}$ and $b \in [1, 2]$.

- (i) If $0 \leq a(n) \leq h(n) - 2$ or $b \geq \frac{5}{4} + \Omega(1)$, then $\mathbb{E}(X) \gg \mathbb{E}(X')$.
- (ii) If $h(n) - 1 \leq a(n)$ and $b \leq \frac{5}{4} - \Omega(1)$, then $\mathbb{E}(X) \ll \mathbb{E}(X')$.

Proof. If $a = o(n)$, then we have $\mathbf{E}(X') = \Theta(2^{L/2})$ by Theorem 5.1. Only level i' is significant for $\mathbf{E}(X')$.

If $a \geq 0$ is very small, say $a = |O(\log \log n)|$, then by Theorem 3.8 (i),

$$\frac{\mathbf{E}(X_{i_{\delta'}})}{2^L} \geq \frac{2^{-O(\log \log n)}}{e^{2^{O(\log \log n)}}} \geq \frac{2^{-O(\log \log n)}}{e^{(\log n)^{O(1)}}} = e^{-o(n)} = 2^{-o(L)},$$

so $\mathbf{E}(X) = \Omega(2^{L(1+o(1))}) \gg \mathbf{E}(X')$.

If $a \in [1 .. h - 2]$ and $a \rightarrow +\infty$, then $\mathbf{E}(X) = 2^{\tilde{c}'n}$ by Theorem 6.1 (ii), where $\tilde{c}' \geq 1 - 2^{-2} \log e + o(1) \doteq 0.639 > 1/2$, and we still have $\mathbf{E}(X') = \Theta(2^{L/2})$. So again, $\mathbf{E}(X) \gg \mathbf{E}(X')$.

If $a = h - 1$, then $\mathbf{E}(X) = 2^{1-\log e/2+o(1)} \ll \mathbf{E}(X')$ by Theorem 6.1 (ii).

For $a \geq h$, we invoke Theorem 6.2. If $b \leq \frac{5}{4} - \Omega(1)$, then $2 - \frac{2}{b} < \frac{1}{2b}$, so $\mathbf{E}(X) \ll \mathbf{E}(X')$. If $b \geq \frac{5}{4} + \Omega(1)$, then $2 - \frac{2}{b} > 1 - \frac{1}{b} > \frac{1}{2b}$, so $\mathbf{E}(X) \gg \mathbf{E}(X')$. \square

6.2 Another Kind of Large Deviation Inequalities

A remark on large deviations is appropriate. For several parametrisations of n , the cut-off point $2n2^{\frac{1+c}{2}L}$ from Corollary 5.6 is much larger than $\mathbf{E}(X)$ or $\mathbf{E}(X')$, so we cannot derive meaningful large deviation results in the optimal variable ordering case. For example, we can only say that $X_* = O(2^{\frac{1+o(1)}{2}L})$ with (very) high probability for the parametrisation $n = 2^h + 2h$, whereas Theorem 6.1 (i) asserts that $\mathbf{E}(X) = n^{2+o(1)}$.

For the case of a *fixed* variable ordering, one can use Markov's inequality straightforwardly to derive (rather weak) concentration inequalities. We omit the trivial details.

Proposition 6.5. (Markov's Inequality) For any nonnegative random variable R ,

$$\Pr\left(R \geq \lambda \cdot \mathbf{E}(R)\right) \leq \frac{1}{\lambda}. \quad \square$$

For example, if $n = 2^h + 2h$, then $\Pr(X = n^{2+\Omega(1)}) = o(1)$.

But we can also prove something for *optimal* variable orderings, if we are willing to consider 'really large' deviations.

Theorem 6.6. For $R \equiv X$ or $R \equiv X'$ and large n , we have

$$\Pr\left(R_* \geq \lambda \cdot 2^{4 \log^2 n} \cdot \mathbf{E}(R)\right) \leq \frac{1}{\lambda}.$$

Proof. Clearly,

$$R_* = \max_{\pi} R_{\pi} \leq \sum_{\pi} R_{\pi}, \quad (6.6)$$

where π runs over all variable orderings. The idea is to exploit the fact that almost all reductions take place at levels near the bottom to reduce the number of π in the sum. Actually, we will use a more refined version of (6.6), namely

$$R_* = \max_{\pi} R_{\pi} = \max_{\pi} \sum_{j \in [n]} R_{\pi, j} \leq \max_{\pi} \sum_{j=1}^{j'-1} R_{\pi, j} + \max_{\pi} \sum_{j=j'}^n R_{\pi, j}, \quad (6.6')$$

where $j' := \lfloor i_0 \rfloor - 2 \log n$. The key observation (already used by Wegener [Weg94]) is that $R_{\pi, j}$ does not depend on the relative order of the first (with respect to π) $j - 1$ variables and the last $n - j$ variables. Therefore,

$$\max_{\pi} \sum_{j=j'}^n R_{\pi, j} = \max_{\pi \in \Pi} \sum_{j=j'}^n R_{\pi, j},$$

where Π is a set of representatives of size

$$\#\Pi = \binom{n}{j'-1} j' \leq \binom{n}{n-j'+1} n = o(2^{4 \log^2 n})$$

– we choose a $j' - 1$ variable set to be tested above level j' , and the variable which is tested at level j' . So

$$\mathbb{E}(R_*) \leq n! \sum_{j=1}^{j'-1} \mathbb{E}(R_j) + o(2^{4 \log^2 n} \mathbb{E}(R))$$

by (6.6'). To prove that the sum over the upper levels is negligible, we refer to the preceding analyses.

In the case $R \equiv X$, for $j \leq j'$ we have $\mathbb{E}(X_j) \leq 2^{-n \log n} / n$ by Lemma 3.9. So $n! \sum_{j=1}^{j'-1} \mathbb{E}(X_j) = o(1)$.

In the case $R \equiv X'$, for $j \leq j'$ we have $\mathbb{E}(X'_j) \leq 2^{-\Omega(L^3)}$ by Theorem 5.1 (i). So again, $n! \sum_{j=1}^{j'-1} \mathbb{E}(X'_j) = o(1)$.

We have shown that in both cases, $\mathbb{E}(R_*) \leq 2^{4 \log^2 n} \mathbb{E}(R)$ for large n . Now the theorem follows by Markov's inequality (Proposition 6.5). \square

Since $R_* \geq R$, we obtain the following large deviation result.

Corollary 6.7. Let $n \rightarrow +\infty$ and $R \equiv X$ or $R \equiv X'$. Then

$$\Pr \left(\frac{R_*}{\mathbb{E}(R_*)} = \omega \left(2^{4 \log^2 n} \right) \right) = o(1). \quad \square$$

How good is this bound? In the cases where $R = 2^{\Omega(n)}$, we determined $E(R)$ up to a factor of $2^{o(1)}$, so these bounds hold essentially unchanged for R_* with high probability, too. Only for $R \equiv X$ it can happen that $E(X) = O(2^{4 \log^2 n})$ — namely if $a(n) \in [h(n) .. 2h(n)^2(1+o(1))]$, which is a fairly small, but interesting, interval (see Theorem 6.1 (i)). The factor 4 in the exponent in Corollary 6.7 could be improved to 1 using a somewhat more complicated technique which will be developed in Section 10, but not to $o(1)$.

Let us conclude with the remark that we know how large X_* and X'_* are with high probability, and their expected values, up to a factor of $2^{o(n)}$, which is very small in comparison with the expected size of an optimal OBDD for a random Boolean function ($2^{(1+o(1))n}$) and the expected amount of reduction achievable by the deletion rule ($\geq 2^{(1/3+o(1))n}$).

7 Other Decision Diagrams with a Variable Ordering

7.1 Zero-Suppressed Binary Decision Diagrams (ZBDDs)

It has been observed by Löbbing, Schröder, and Wegener [SW98a,LSW95] that zero-suppressed BDDs behave quite similar to OBDDs, if random Boolean functions are considered. The analyses of Liaw and Lin [LL92] and Wegener [Weg94] carry over without major changes. This is true for our results, too.

First, observe that a quasireduced ZBDD is the same as a qOBDD, because both binary decision diagram types use the same merging rule. Therefore, the analysis of qOBDDs we gave in Sections 3 and 4 does not need to be modified.

The key observation is that the modified deletion rule in ZBDDs leads to the same probability distribution of X'_i . We quote from the proof of Theorem 5.1, page 59: “Among the m_i subfunctions which are possible at level i of the qOBDD there are m_{i+1} functions that do not depend essentially on the variable x_i .” For ZBDDs, this sentence should read: “Among the m_i subfunctions which are possible at level i of the qZBDD = qOBDD there are m_{i+1} functions g such that $g_{x_i=1} = 0$.” For random $g = g(x_i, \dots, x_n)$, the events $g(0, x_i, \dots, x_n) = g(1, x_i, \dots, x_n)$ and $g(1, x_i, \dots, x_n) = 0$ have the same probabilities, since we consider the uniform distribution.

All results of Sections 5 and 6 hold for ZBDDs, too. In particular, we have a modified ‘Main Theorem 1’.

7.2 Ordered Kronecker Functional Decision Diagrams (OKFDDs)

The situation is slightly more complicated for OKFDDs, because the choice of the decomposition type list constitutes another potential for minimisation.

Again, quasireduced OKFDDs are the same as qOBDDs. Also, regardless which decomposition type is performed at a level, a node can be deleted if and only if the subfunction it represents does not depend on the variable tested there, so we do not even need to modify the proof of Theorem 5.1 as for ZBDDs.

If the OKFDD is in fact an OFDD, then the same conclusions as for ZBDDs can be made.

For arbitrary decomposition type lists, the key observation is that multiplying the doubly exponential probability bounds with either $n!$ or $3^n n!$ does little harm, see (5.11’) on page 65. Thus, the large deviation results from Section 5 hold for OKFDDs as well (only the $o(1)$ terms change). This includes the *weak* Shannon effect (Corollary 5.6) and a ‘Main Theorem 1’ for OKFDDs.

We remark that there seems to be no way to overcome this difficulty in the framework of [Weg94] due to the weaker probability bounds. The proof of Theorem 6.6 (which is based on Markov’s inequality) does not carry over to OKFDDs. The results of Section 6.1 hold for each decomposition type list.

PART 2:

RANDOM FBDDs

8 Minimal FBDDs

This section contains preliminary considerations on minimal FBDDs. After fixing some terminology, we show that the class of FBDDs in which the nodes represent ‘different’ subfunctions (in a sense to be defined below) is characterised by *strong* versions of the standard reduction rules. Since minimal FBDDs are ‘strongly reduced’, they have this property, which will be used in afterwards. See Section 9 for an outline of the proof of Main Theorem 2.

8.1 Definitions

The *support* of a Boolean function is the set of variables it depends on. Usually, the support is clear from the context, and although formally incorrect, Boolean functions like

$$f(x_1, x_2) = x_1 \quad \text{and} \quad g(x_1) = x_1 \quad (8.1)$$

are taken as if they were the same. — We will say that f and g are *equivalent*, but *not equal* subfunctions. We admit that this distinction is a sort of ‘abstract nonsense’ if considered on its own, but it enables us to state and prove some observations not possible otherwise. — In the following, we assume that f is a Boolean function with support $\Xi := \{x_1, \dots, x_n\}$. Sometimes ξ is used as a metavariable for the elements of Ξ .

We can specify a node in an FBDD (like in any ‘-DD’) by declaring the values of the variables tested along a ‘partial’ computation path leading to it. The corresponding concept for Boolean functions are *partial assignments*. In a partial assignment, there can be variables which are not assigned values. These are called *free*. If we impose a partial assignment on a Boolean function, we obtain a *cofactor*.

Definition 8.1.

- (i) A *partial assignment* is a mapping $\alpha : \Xi \rightarrow \mathfrak{3} = \{0, 1, 2\}$.
- (ii) The set of *free* variables of a partial assignment is defined by $\text{free}(\alpha) := \alpha^{-1}\{2\}$, and its complement is $\overline{\text{free}}(\alpha) := \Xi \setminus \text{free}(\alpha) = \alpha^{-1}\{0, 1\}$.
- (iii) The *cofactor* of a Boolean function f for a partial assignment α is a Boolean function f_α of the variables in $\text{free}(\alpha)$. The value of f_α for an argument $c = (c_i \mid x_i \in \text{free}(\alpha)) \in {}^{\text{free}(\alpha)}2$ is defined by $f_\alpha(c) := f(c^\alpha)$, where $c^\alpha := (c_1^\alpha, \dots, c_n^\alpha)$ and

$$c_i^\alpha := \begin{cases} c_i, & \alpha(x_i) = 2; \\ \alpha(x_i), & \text{otherwise.} \end{cases}$$

- (iv) Two partial assignments α and β are *equivalent* for a Boolean function f (written $\alpha \approx_f \beta$, or simply $\alpha \approx \beta$, if f is clear from the context), if

$$\forall c = (c_1, \dots, c_n): f(c^\alpha) = f(c^\beta).$$

The simplest example where cofactors occur is the Shannon decomposition

$$f(x_1, x_2, \dots, x_n) = \bar{x}_1 \wedge f(0, x_2, \dots, x_n) \vee x_1 \wedge f(1, x_2, \dots, x_n).$$

Note that $\alpha \approx_f \beta$ does *not* imply that $\text{free}(\alpha) = \text{free}(\beta)$. The functions f and g from (8.1) may serve as an example.

In an FBDD, many nodes are reached along different (partial) computation paths, that is, the corresponding cofactors are equivalent. Intuitively, a *subfunction* of f is just “what is represented at some internal node of an FBDD for f ”. But then a difficulty arises, because in terms of FBDDs, there is also a natural notion what the support of a subfunction should be (namely, the set of variables tested somewhere below the subfunction node), and this set need not be the same as the set of variables which have not been tested yet. There might be some variables which are not tested at all. This ambiguity is the reason why we have to distinguish between *equivalence* and *equality* of Boolean functions.

Definition 8.2.

- (i) A *subfunction* of a Boolean function f is a cofactor f_α of f for some partial assignment α . A subfunction f_α is said to *depend* on a variable x_i , if $x_i \in \text{free}(\alpha)$. By convention, subfunctions “do not look at” variables on which they do not depend, and we will write $f_\alpha(c) := f_\alpha((c_i \mid x_i \in \alpha^{-1}\{2\}))$ to simplify the notation. As a special case, we define $f_{\bar{x}_i} := f_{\xi \mapsto 0}$ and $f_{x_i} := f_{\xi \mapsto 1}$, where $x_i \mapsto c$ denotes the partial assignment defined by $(x_i \mapsto c)(x_i) := c$, and $(x_i \mapsto c)(\xi) = 2$ for $\xi \in \Xi \setminus \{x_i\}$.
- (ii) Two subfunctions f_α and f_β of f are said to be *equivalent*, if $\alpha \approx_f \beta$ (because then, $f_\alpha(c)$ and $f_\beta(c)$ take ‘equal values’ for all c), and *equal* (written $f_\alpha = f_\beta$), if $\alpha \approx_f \beta$ and $\text{free}(\alpha) = \text{free}(\beta)$ (because then they are equal as functions, thinking of their free variables as argument positions). Functions are said to be *different* if they are not equal. Also, we say that they are the *same* if they are equal.

Stated in another way, the difficulty is that cofactors with different supports may correspond to the same FBDD node.

Our notion of dependency allows that a function may depend on a variable whose value need not be known to determine its value. That is, we can have $f_{\alpha, \bar{x}_i} = f_{\alpha, x_i}$ and $x_i \in \text{free}(\alpha)$. Those variables whose value must be known are called *essential*.

Definition 8.3. A variable ξ is *essential* for a Boolean function f , if $f_{\bar{\xi}} \not\approx f_{\xi}$. We denote the set of essential variables of f by $\text{ess}(f)$.

We fix some notation: F, F', D, D' , and V .

Definition 8.4. Let v be a node of an FBDD for a Boolean function f . Then v represents a subfunction $F(v)$ of f , which by definition depends on $D(v)$, the set of variables tested below v . The essential variables set of $F(v)$ is $D'(v) := D(v) \cap \text{ess}(F(v))$. The function depending on $D'(v)$ which is equivalent to $F(v)$ is denoted by $F'(v)$. Let $V(v)$ denote the variable tested at v .

Note that $F(v) \approx F(w)$ if and only if $F'(v) = F'(w)$. Also, $F(v) = F(w)$ implies that $D(v) = D(w)$.

8.2 Strong Reduction Rules

We are mainly interested in *minimal* FBDDs, i. e., those of minimal *size*.

Definition 8.5. Let G be an FBDD representing a Boolean function f . Then $\text{nodes}(G)$ denotes its node set (including the sinks) and $\text{size}(G) := \#\text{nodes}(G)$. We say that G is *minimal*, if $\text{size}(G)$ is minimal among all FBDDs representing f .

Sometimes the terminal nodes are counted in the size, but the difference is asymptotically negligible.

Minimal FBDDs should not be confused with reduced ones. Of course, a minimal FBDD is reduced. But in the absence of a global variable ordering, the standard reduction rules are no longer sufficient to ensure that the nodes of a reduced FBDD represent different (or even nonequivalent) subfunctions, and this property is necessary for being minimal. See Fig. 13 on page 85 for an example.

Since our analysis relies on the property that the nodes represent different subfunctions, we introduce *strongly reduced* FBDDs as an intermediate class. Strongly reduced FBDDs can be characterised by the strong merging rule and the strong deletion rule, which are defined next.

We feel that the strong reduction rules are a very natural and self-suggesting concept and hence do *not* claim originality.

Definition 8.6. An FBDD is *strongly reduced*, if neither the *strong merging rule* M^+ nor the *strong deletion rule* D^+ is applicable to its nodes.

M^+ : Two nodes v and v' can be *merged*, if they represent the same subfunction $F(v) = F(v')$.

D^+ : A node v can be *deleted*, if $V(v)$ is not essential for $F(v)$, that is, if $V(v) \notin D'(v)$.

The choice which node to keep is free in both cases.

The reader is invited to check that M^+ and D^+ indeed transform an FBDD into a smaller FBDD representing an equivalent function.

Strongly reduced FBDDs are not necessarily minimal. For example, a reduced OBDD is always strongly reduced, and need not be minimal. The next lemma shows that the strong reduction rules characterise the class of FBDDs in which the nodes represent nonequivalent subfunctions.

Lemma 8.7. An FBDD is strongly reduced if and only if its nodes represent non-equivalent subfunctions.

Proof. Assume that v and v' are two nodes of an FBDD which represent equivalent subfunctions $F(v) \approx F(v')$. If $D(v) = D(v')$, then $F(v) = F(v')$ and M^+ is applicable. If $D(v) \neq D(v')$ holds, then their symmetric difference $D(v) \Delta D(v') := D(v) \setminus D(v') \cup D(v') \setminus D(v)$ is non-empty. Let $\xi \in D(v) \Delta D(v')$, without loss of generality $\xi \in D(v)$, and let w be a node below v where ξ is tested. Since $\xi \notin D(v')$, we have $\xi \notin D'(v) = D'(v') \subseteq D(v')$, and since $F(w)$ is a subfunction of $F(v)$, we have $D'(w) \subseteq D'(v) \not\ni \xi = V(w)$. So D^+ is applicable to w . This shows that strongly reduced FBDDs have non-equivalent subfunctions.

The converse implication is easy. If M^+ is applicable to v and w , then $F(v) = F(w)$, so $F(v) \approx F(w)$. If D^+ is applicable to u , and u' is one of its successors, then $F(u) \approx F(u')$. \square

We observe that strongly reduced FBDDs do not perform ‘redundant’ tests like the test for x_1 in Fig. 13. Note that in this example, the strong deletion rule is applicable to the source, and that there is indeed a choice which node to keep after deletion.

Stated in another way, the fine distinctions between D and D' and between F and F' we made in Definition 8.4 do not apply in strongly reduced (or even minimal) FBDDs.

Corollary 8.8. In a strongly reduced FBDD, we have $D(v) = D'(v)$ and $F(v) = F'(v)$ for all nodes v .

Proof. Let $\xi \in D(v) \setminus D'(v)$ be a non-essential variable of $F(v)$. Then there is a node w below v with $V(w) = \xi$, and $V(w)$ is non-essential for $F(w)$, because $F(w)$ is a subfunction of $F(v)$. But this implies that w and its successor node(s) represent equivalent subfunctions, a contradiction. Clearly, $D(v) = D'(v)$ implies $F(v) = F'(v)$. \square

The strong reduction rules are very different from the standard ones from an algorithmic point of view. Note that the strong reduction rules M^+ and D^+ are defined using the *semantic* notions ‘equal’ and ‘essential’, which cannot be read off from the FBDD using local structural information. At this time, it is open (but not an important issue in our context) whether the strong reduction rules have an efficient algorithm, like the ‘standard’ reductions. An efficient algorithm for the test $F(v) = F(v')$ required in M^+ would in particular enable us to check the equivalence of FBDDs, a problem for which only a randomised algorithm with one-sided error is known so far [BCW80]. There is a trivial gadget (Fig. 14) showing that the test $V(v) \in D'(v)$ required for D^+ is of the same degree of difficulty.

While a lot of (mostly experimental) work has been done on OBDD reordering techniques, very little is known about FBDD minimisation techniques so far (but see [BMS96,GD99]). Using the additional potential for reduction seems to be difficult in practice, due to the vast size of the search space.

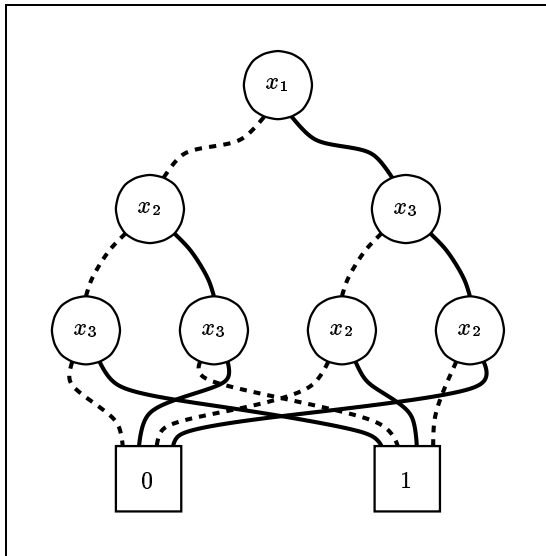


Fig. 13. A reduced FBDD for $x_2 \oplus x_3$ with a redundant test for x_1

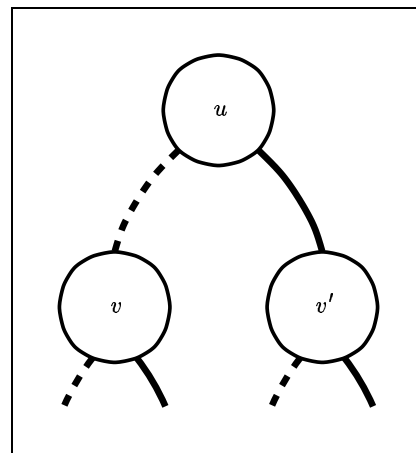


Fig. 14. $F(v) \approx F(v') \leftrightarrow V(u) \notin D'(u)$

9 Quasireduced FBDDs and the Deletion Rule

To estimate the size of minimal FBDDs, we will investigate the effect of the two reduction rules in turn, as we did for OBDDs. This section is devoted to the deletion rule.

If only the merging rule is applied to a complete binary decision tree in which Shannon decomposition is performed with respect to some variable ordering, then the result is a qOBDD. In the absence of a global variable ordering, the result is what we call a quasireduced FBDD, or qFBDD for short.

Definition 9.1. An FBDD is *quasireduced*, if it is reduced with respect to the merging rule and every variable is tested along every computation path. The minimal qFBDD size of a Boolean function f is denoted by $\Psi'(f)$.

Recall that the minimal FBDD size was given the name Ψ in Main Theorem 2. Quasireduced FBDDs will serve us as an intermediate step in the analysis of the reduction process. In this section, we show that for *most* Boolean functions, *every* strongly reduced FBDD can be converted into a qFBDD by adding only a comparatively small number of new nodes using the *inverse deletion rule*. The inverse deletion rule is just the ordinary deletion rule applied in the reverse direction.

As a consequence, the minimal FBDD size of a random Boolean function is almost equal to its minimal qFBDD size with high probability. More precisely, in Corollary 9.7 we give a doubly exponential probability bound for deviations of $\Psi' - \Psi$ which are not much larger than those we considered for OBDDs in Section 5. Actually, this bound follows from a slightly more general result on the effect of the algorithm INVERSE-DELETION (which is described next) on strongly reduced FBDDs.

Therefore, later sections can deal with Ψ' instead of Ψ . Section 10 is devoted to lower bounds on Ψ' , leading to Main Theorem 2 (i). In Section 11, an upper bound for Ψ' is derived by an explicit (algorithmic) construction, proving Main Theorem 2 (ii).

9.1 Algorithm INVERSEDELETION

Algorithm INVERSEDELETION will serve us to estimate the effect of the deletion rule. An example application is shown in Fig. 15. There is a technical reason why our analysis of the reduction rules proceeds in the reverse direction for FBDDs: To prove that Ψ' is not much larger than Ψ , we show that every minimal FBDD can be converted into a qFBDD by relatively few applications of the inverse deletion rule. Let $G' := \text{INVERSE-DELETION}(G)$. If G was reduced, then G' will be quasireduced, but we can estimate $\text{size}(G') - \text{size}(G)$ only if G was strongly reduced. Simply blowing up the FBDD in an arbitrary way does not work; to bound the number of inserted nodes, we need

the property that in the resulting qFBDD, all nodes represent *different* subfunctions, in the sense defined in Section 8. Otherwise, for example the node marked with “!” in Fig. 15 could have been generated twice, wrecking our argumentation. The output of INVERSEDELETION has this property, if the input was strongly reduced (Lemma 9.4), and later we will apply INVERSEDELETION to minimal FBDDs.

For those readers who are acquainted with graph driven BDDs, we mention that INVERSEDELETION is very similar to the algorithm of [SW95, Theorem 1] that constructs an oracle graph G_0 from a read-once branching program G such that G is a weak G_0 -FBDD (which was called G_0 -driven WBDD there). The main difference is the local merging step, which is necessary for Lemma 9.4.

Definition 9.2. For a node v of an FBDD, we denote the set of its (immediate) predecessor nodes by $P(v)$ and the set of variables tested along any computation path going through v before it leaves v by $T(v)$. So in particular, $V(v) \in T(v)$. The ‘top’ node of an FBDD G is denoted by $\text{source}(G)$.

Algorithm INVERSEDELETION

Input: A reduced FBDD G .

Output: A quasireduced FBDD G' , equivalent to G .

- Each node can carry a ‘mark’. Initially, all nodes are unmarked.
An unmarked node becomes ‘active’, if all its predecessors are marked.
- Using the inverse deletion rule, insert a chain of new nodes in front of $\text{source}(G)$, in which the variables in $\Xi \setminus D(\text{source}(G))$ are tested, respecting the canonical variable ordering.
- Mark $\text{source}(G)$.
- Repeat the following steps, until both sinks have been marked:
 - Let v be an active node.
 - For all predecessors $u \in P(v)$ of v :
Using the inverse deletion rule, replace the edge uv by a chain of nodes, in which all variables in $T(v) \setminus T(u) \setminus V(v)$ are tested, respecting the canonical variable ordering.
 - Apply the merging rule to the nodes inserted during the last step.
 - Mark v .

Proposition 9.3. Assume that u and v were inserted by INVERSEDELETION while w was active, and not merged before w was marked. Then $D(u) \neq D(v)$.

Proof. Assume that u and v were inserted while w was active, and $D(u) = D(v)$. To prove that u and v were merged, we apply induction on $z := \#D(u) \setminus D(w) = \#D(v) \setminus D(w)$. Since u and v were inserted respecting the canonical variable ordering and $D(u) = D(v)$, we must have $V(u) = V(v)$. Let u' and v' denote the successor nodes of u and v , respectively. Now either $z = 1$, which implies $u' = w = v'$, or the variable ordering implies that $V(u') = V(v')$, and u' and v' have been merged by the induction hypothesis. So the merging rule is applicable to u and v . \square

Lemma 9.4. If G is a strongly reduced FBDD and $G' := \text{INVERSEDELETION}(G)$, then the nodes of G' represent *different* subfunctions.

Proof. Consider a node v that was inserted while v' was active and let $w \neq v$ be another node of G' . We must show that $F(v) \neq F(w)$. If $w \neq v'$ and w was already present in G , then we have $F(w) \not\approx F(v')$ by Lemma 8.7. Since $F(v) \approx F(v')$, we have $F(w) \not\approx F(v)$, which implies $F(w) \neq F(v)$. If $w = v'$, then $F(w) \neq F(v)$ because $D(w) = D(v') \neq D(v)$. So assume that w was inserted while w' was active. If $w' \neq v'$, then $F(v) \neq F(w)$ because $F(v) \approx F(v') \not\approx F(w') \approx F(w)$ (again using Lemma 8.7). If $w' = v'$, then $D(w) \neq D(v)$, because otherwise by Proposition 9.3, w and v would have been merged before $w' = v'$ was marked. So $F(w) \neq F(v)$. \square

To bound the number of insertions during INVERSEDELETION, we give an upper bound on the number of subfunctions which are represented at nodes which *can* be inserted while running INVERSEDELETION on *any* FBDD for a particular function f . This is possible because each such node v corresponds to an ordered pair (f_α, ξ) consisting of a subfunction f_α of f having nonessential variables in its support and one of these nonessential variables $\xi = V(v)$, which is tested at the node v (see the definition of $\text{Del}(f)$ below). Using Lemma 9.4, it is easy to see that for each such pair at most *one* node is inserted during a run of INVERSEDELETION. Thus, we obtain an inequality.

Corollary 9.5. Let f be a Boolean function and $\text{del}(f) := \# \text{Del}(f)$, where

$$\text{Del}(f) := \{(f_\alpha, \xi) \mid \alpha \in \exists 3 \wedge \xi \in \text{free}(\alpha) \setminus \text{ess}(f_\alpha)\}.$$

If G is a strongly reduced FBDD for f and $G' = \text{INVERSEDELETION}(G)$, then

$$\text{size}(G') - \text{size}(G) \leq \text{del}(f).$$

Proof. We show that the mapping

$$\varphi: \text{nodes}(G') \setminus \text{nodes}(G) \rightarrow \text{Del}(f); v \mapsto (F(v), V(v))$$

is injective. Note that $(F(v), V(v)) \in \text{Del}(f)$, because the deletion rule is applicable to v . The mapping φ is injective, since by Lemma 9.4, already $v \mapsto F(v)$ is injective. \square

We hope that the reader will agree with us that the subtleties of Section 8 have payed off by now.

The main ideas leading to Corollary 9.5 can be found in [Weg94, proof of Theorem 7]. However, Wegener's explanation of the unfolding process is *very* short and we cannot follow his argument because the problem with multiple FBDD nodes corresponding to the same application of the deletion rule in a qOBDD is not even mentioned. We understood that he does not exclude that the node marked “!” in Fig. 15 could be generated twice.

9.3 Probabilistic Analysis of INVERSEDELETION

Now we switch over to probabilities. Using results from the analysis of the deletion rule in the OBDD case, we prove a large deviation inequality for del . By Corollary 9.5, this also bounds the number of nodes which can be inserted by INVERSEDELETION.

Theorem 9.6. For every $c > 0$,

$$\Pr_f \left(\text{del}(f) \geq n 2^{\left(\frac{1}{2} + c\right)n} \right) \leq 2^{-2^{(1+o(1))cn/2 \log n}}.$$

Proof. We work with the ‘stratification’ $\text{Del}(f) = \bigcup_{j \in [n]} \text{Del}(f, j)$, where

$$\text{Del}(f, j) := \{(f_\alpha, \xi) \mid \alpha \in \exists 3 \wedge \xi \in \text{free}(\alpha) \setminus \text{ess}(f_\alpha) \wedge \#\overline{\text{free}}(\alpha) = j - 1\}.$$

If $(f_\alpha, \xi) \in \text{Del}(f, j)$, then $f_{\alpha, \xi} = f_{\alpha, \bar{\xi}}$, and this corresponds to the deletion of an FBDD node which is reached along the partial computation path determined by the partial assignment α . On the other hand, there is a unique variable ordering π that satisfies $\overline{\text{free}}(\alpha) <_\pi \xi <_\pi \text{free}(\alpha) \setminus \{\xi\}$ and coincides with the canonical variable ordering within both $\overline{\text{free}}(\alpha)$ and $\text{free}(\alpha) \setminus \{\xi\}$, and (f_α, ξ) corresponds to an application of the deletion rule at the j -th level (where ξ is tested) of $\text{qOBDD}_\pi(f)$. In this way, we can apply the results on the effect of the deletion rule in qOBDDs from Section 5 to estimate $\text{del}(f, j) := \#\text{Del}(f, j)$. To get an upper bound for $\text{del}(f, j)$, we sum up $X'_{\pi, j}(f)$ over all variable orderings π of the form described above. We consider two ranges of j separately.

If $j \geq (1 - \frac{c}{2 \log n})n + 1$, then there are at most

$$\binom{n}{j} j = \binom{n}{n-j} j \leq n^{\frac{c}{2 \log n} n} = 2^{\frac{c}{2} n}$$

ways to choose π . Therefore,

$$\Pr_f \left(\text{del}(f, j) \geq 2^{(\frac{1}{2} + c)n} \right) \leq 2^{\frac{c}{2} n} e^{-(2+o(1))2^{cn}} = e^{-(2+o(1))2^{cn}}$$

by Theorem 5.4. (For every variable ordering π , $X'_{\pi, j}$ is small with high probability.)

For $j \leq (1 - \frac{c}{2 \log n})n + 1$, we apply Markov's inequality (see Proposition 6.5).

$$\begin{aligned} \Pr_f \left(\text{del}(f, j) \geq 1 \right) &= \Pr \left(\sum_{\pi} X'_{\pi, j} \geq 1 \right) \\ &\leq \mathbb{E} \left(\sum_{\pi} X'_{\pi, j} \right) = \sum_{\pi} \mathbb{E}(X'_{\pi, j}) \leq n! \mathbb{E}(X'_{\pi, j}). \end{aligned}$$

Writing $j = i_\delta$, we have

$$\delta = j - i_0 = \left(1 - \frac{c}{2 \log n}\right)n - (1 + o(1))n \sim \frac{-cn}{2 \log n},$$

and by Theorem 5.1,

$$\mathbb{E}(X'_{\pi, j}) \leq (1 + o(1))2^{\delta + (1 - 2^{-\delta - 1})L} = 2^{-2^{(1+o(1))cn/2 \log n}}.$$

So

$$\Pr_f \left(\text{del}(f, j) \geq 1 \right) \leq n! 2^{-2^{(1+o(1))cn/2 \log n}} = 2^{-2^{(1+o(1))cn/2 \log n}},$$

and the theorem follows by summing over all levels. \square

Corollary 9.7. For every $c > 0$,

$$\Pr \left(\Psi' - \Psi \geq n 2^{\left(\frac{1}{2}+c\right)n} \right) \leq 2^{-2^{(1+o(1))cn/2 \log n}},$$

and for almost all Boolean functions, minimal quasireduced FBDDs and minimal FBDDs have almost the same size.

Proof. Let G be an FBDD for a Boolean function f with $\text{size}(G) = \Psi(f)$. Since G is minimal, it is also strongly reduced, and by Corollary 9.5 we have $\text{size}(G') - \text{size}(G) \leq \text{del}(F)$, where $G' := \text{INVERSEDELETION}(G)$. Thus, $\Psi'(f) \leq \text{size}(G') \leq \Psi(f) + \text{del}(F)$, and $\text{del}(F)$ is ‘small’ with high probability, see Theorem 9.6.

The second claim follows from a result of Breitbart, Hunt, and Rosenkrantz who showed that almost all Boolean functions have a minimal BP size of $(1 + o(1))2^n/n$, which is a lower bound for Ψ [BHR95, page 55]. \square

10 FBDDs and the Merging Rule

The aim of this section is to prove Main Theorem 2 (i). So far, we know that $\Psi' \sim \Psi$ with high probability (Corollary 9.7). Using the *inverse merging rule*, a qFBDD can be unfolded into a binary decision tree. This is done by Algorithm INVERSEMERGING, which is described below. Since we know the number of nodes in a complete binary decision tree, we can determine the size of the original qFBDD if we know (at least approximately) the number of nodes which are inserted by INVERSEMERGING. There are some technical complications in the ‘lower’ part of the qFBDD, however.

For quasireduced FBDDs, there is a natural notion of a *level* that generalises the notion of a level in qOBDDs. Levels are determined by the distance from the source. Mergings are only possible within levels.

Definition 10.1. Let G be a qFBDD and $j \in [n]$. Then

$$\text{Level}(G, j) := \{v \in \text{nodes}(G) \mid \#D(v) = n - j + 1\}$$

and $\text{level}(G, j) := \#\text{Level}(G, j)$.

The lower part of an qFBDD consists of the levels below i_0 . Fortunately, it turns out that the lower part has only comparatively few nodes under the assumptions of Main Theorem 2 (i).

The main result of this section is Theorem 10.10, which implies Main Theorem 2 (i).

10.1 Algorithm INVERSEMERGING

Algorithm INVERSEMERGING is very simple, and stated here mainly for reference.

Algorithm INVERSEMERGING

Input: A quasireduced FBDD G .

Output: A complete binary decision tree G' equivalent to G .

- For $j = 2, \dots, n$:
 - Expand the nodes at level j using the inverse merging rule.
- Expand the terminals using the inverse merging rule.

It is straightforward to implement INVERSEMERGING using $O(2^n)$ time and space. The decision tree can be output during a ‘lexicographical’ traversal of the qOBDD (with respect to the levels, not the variables). This is optimal due to the size of the result.

10.2 Relating qFBDDs to qOBDDs

Assume that G is a minimal qFBDD for some Boolean function f and $G' := \text{INVERSE-MERGING}(G)$. We want to lower bound $\text{size}(G)$. Using the results from the qOBDD case, we will derive upper bounds on $\text{level}(G', j) - \text{level}(G, j) = k_j - \text{level}(G, j)$ for each level j . The intuition of these estimations is that qFBDDs are smaller than qOBDDs because they can somehow ‘collect’ mergings which are possible within qOBDDs for several variable orderings. But it is *not* necessary to sum up over all $n!$ variable orderings to get an upper bound. To explain why, we need some definitions.

First, let us fix what kind of mathematical entity a variable ordering should ‘really’ be. (We have been successful in avoiding this question so far, but now we will work with variable orderings more specifically.)

Definition 10.2. A *variable ordering* is a bijection $\pi: \mathcal{E} \rightarrow [n]$. We write $\xi <_{\pi} \xi' :\leftrightarrow \pi(\xi) < \pi(\xi')$.

In qOBDDs, the nodes of each level have the same dependency sets. In qFBDDs, we only know that their dependency sets have the same size. Thus, we further partition the nodes of each level into classes with equal dependency sets d . Each dependency set d corresponds to a certain variable ordering π_d . This forges the link to OBDDs.

Definition 10.3. Let $j \in [n]$.

- (i) The set of all dependency sets $D(v)$ which are possible for a node v at level j of a qFBDD is denoted by

$$\Delta(j) := \{d \mid d \subseteq \mathcal{E} \wedge \#d = n - j + 1\}.$$

- (ii) For each $d \subseteq \mathcal{E}$, denote by π_d the unique variable ordering that satisfies $\mathcal{E} \setminus d <_{\pi_d} d$ and coincides with the canonical ordering within both $\mathcal{E} \setminus d$ and d .

Note that $D(v) = \pi_{D(v)}^{-1}[j .. n]$ by definition. This gives us a handy way to get from an application of the merging rule in an FBDD to a corresponding application in an OBDD. It turns out that the order in which the variables in $\mathcal{E} \setminus D(V)$ have been tested in the qFBDD is irrelevant, as is $V(v)$. It is only the sets of variables $D(v)$ which are important. In this way, the number of π which have to be considered can be reduced tremendously. This idea is essentially due to Wegener [Weg94].

We note an easy observation which will be used in the proof of the lemma following it. It relates the size of certain qOBDD levels to the number of subfunctions with prescribed supports.

Proposition 10.4. Let $j \in [n]$ and $d \in \Delta(j)$. Then

$$Y_{\pi_d, j}(f) = \text{level}(\text{qOBDD}_{\pi_d}(f), j) = \#\{f_\alpha \mid \alpha \in \Xi\mathbb{3} \wedge \text{free}(\alpha) = d\}.$$

Proof. We give a ‘proof by bijection’. (The first equality holds by definition.) Let $v \in \text{Level}(\text{qOBDD}_{\pi_d}(f), j)$. Then v represents a subfunction $F(v)$ of f which depends on the variables in $\pi_{D(v)}^{-1}[j..n] = d$. Due to the merging rule, all the $F(v)$ at this level are different. Conversely, each subfunction f_α with $\text{free}(\alpha) = d$ is represented by the node $v \in \text{Level}(\text{qOBDD}_{\pi_d}(f), j)$ which is reached along the (partial) computation path which is determined by the values of α on $\Xi \setminus d$. Therefore,

$$\begin{aligned} \text{level}(\text{qOBDD}_{\pi_d}(f), j) &= \#\{F(v) \mid v \in \text{Level}(\text{qOBDD}_{\pi_d}(f), j)\} \\ &= \#\{f_\alpha \mid v \in \Xi\mathbb{3} \wedge \text{free}(\alpha) = d\}. \quad \square \end{aligned}$$

The next lemma enables us to estimate the gap between expected and worst-case width in the ‘upper’ part of a random qFBDD by using the results on X_j .

Lemma 10.5. Let G be a qFBDD for a Boolean function f and define

$$X_{\Delta, j}(f) := \sum_{d \in \Delta(j)} X_{\pi_d, j}(f).$$

Then for all $j \in [i_0]$,

$$\text{level}(G, j) \geq k_j - X_{\Delta, j}(f).$$

Proof. Let $G' := \text{INVERSEMERGING}(G)$. For any FBDD H and $d \subseteq \Xi$, we denote the set of nodes with dependency set d by $\text{Level}(H, d) := \{v \in \text{nodes}(H) \mid D(v) = d\}$. Also, let $\text{level}(H, d) := \#\text{Level}(H, d)$. Clearly,

$$k_j - \text{level}(G, j) = \sum_{d \in \Delta(j)} (\text{level}(G', d) - \text{level}(G, d)),$$

and due to the merging rule, $\text{level}(G, d) \geq \#\{F(v) \mid v \in \text{Level}(G', d)\}$. (Two nodes can only be merged if they represent the same subfunction. In general, this inequality is strict, because the decomposition variables must also coincide.) There is a bijective

correspondence between each $v \in \text{Level}(G', d)$ and a (partial) assignment to the variables in $\Xi \setminus d$, since G' is a decision tree. Denote this assignment by α_v . Then we have

$$\begin{aligned}
& \text{level}(G', d) - \text{level}(G, d) \\
& \leq \text{level}(G', d) - \#\{F(v) \mid v \in \text{Level}(G', d)\} \\
& = \#\{\alpha_v \mid v \in \text{Level}(G', d)\} - \#\{f_{\alpha_v} \mid v \in \text{Level}(G', d)\} \\
& \leq \#\{\alpha \mid \alpha \in \Xi \mathbf{3} \wedge \text{free}(\alpha) = d\} - \#\{f_\alpha \mid \alpha \in \Xi \mathbf{3} \wedge \text{free}(\alpha) = d\} \\
& = k_j - Y_{\pi_d, j}(f) = X_{\pi_d, j}(f),
\end{aligned}$$

where the second inequality holds simply because $\alpha \mapsto f_\alpha$ is a mapping, and the equality

$$Y_{\pi_d, j}(f) = \#\{f_\alpha \mid \alpha \in \Xi \mathbf{3} \wedge \text{free}(\alpha) = d\}$$

was proved in Proposition 10.4. The lemma follows. \square

The lower bounding technique of Section 10.2 breaks down for levels $j \geq i_0$. Instead, we estimate the worst-case size of the whole lower part. Fortunately, for the range of a in question we have $W^b = o(W)$.

Lemma 10.6. Let $n \rightarrow +\infty$ and denote the worst-case size of all levels below i_0 by

$$W^b := \sum_{j=[i_1]}^n w_j.$$

Then

$$\frac{W^b}{W} = \begin{cases} O(2^{L/\sqrt{2}}), & a \leq 0; \\ \Theta(2^{L-a}), & 0 < a \leq \sqrt{n}; \\ O(2^{L-\sqrt{n}}), & a \geq \sqrt{n}. \end{cases}$$

Proof. By Theorem 2.6, $W = \Theta(2^L)$, and due to the doubly exponential decrease rate of $j \mapsto m_j$ (for each n), we have

$$W^b \sim m_{[i_1]} = m_{i_{[i_1]} - i_0} = 2^{2^{i_0 - [i_1]} L}.$$

Clearly,

$$i_0 - [i_1] = i_0 - [i_0] - 1 = \begin{cases} -\delta' - 1 \leq \frac{-1}{2}, & \delta' < 0; \\ -1, & \delta' = 0; \\ -\delta', & \delta' > 0. \end{cases}$$

This already settles the case $a \leq 0$.

For $0 < a \leq \sqrt{n}$, we have $\tilde{a} = a + O(1)$ by Lemma 3.7 and $2^{-\delta'} L = L - \tilde{a} + O(1)$ by (3.11) and (3.12) from the proof of Theorem 3.8 (i), which implies

$$2^{-\delta'} L = L - a + O(1). \quad (10.1)$$

For $a > \sqrt{n}$, we have $\tilde{a} \geq \sqrt{n} + O(1)$, and

$$2^{-\delta'} L = 2^{-\delta \tilde{a}} L \leq L - \sqrt{n} + O(1)$$

follows from (10.1). \square

We have already seen how the widths of the levels $j \leq i_0$ in a qFBDD are related to those in a qOBDD in Lemma 10.5. Now we estimate this bound for the whole upper part. (The attentive reader will notice a great similarity to the proof of Theorem 6.6.)

Lemma 10.7. Let

$$X_{\Delta}(f) := \sum_{j \in [i_0]} X_{\Delta, j}(f),$$

where $X_{\Delta, j}(f)$ was defined in Lemma 10.5, and

$$\hat{X}(f) := \sum_{j \in [i_0]} X_j(f).$$

Then

$$\mathbf{E}(X_{\Delta}) = o\left(2^{\log^2 n} \mathbf{E}(\hat{X}) + 1\right).$$

Proof. Clearly, $\#\Delta(j) \leq 2^n$. By Lemma 3.9, for $j \in [i_{-4}]$ we have

$$\mathbf{E}(X_j) \leq 2^{\left(1 - \frac{4}{\log e}\right)L} \leq 2^{-2L}$$

for large n . So

$$\sum_{j \in [i_{-4}]} \mathbf{E}(X_{\Delta, j}) = o(1).$$

For $j \in]i_{-4} \dots i_0]$, we have

$$\#\Delta(j) \leq \binom{n}{\log L + 10} \leq \frac{n^{\log L + 10}}{(\log L)!} = o(n^{\log L}) = o(2^{\log^2 n}),$$

because

$$(\log L)! \geq \left(\frac{\log L}{2}\right)^{\frac{\log L}{2}} = 2^{\frac{\log L}{2} \log\left(\frac{\log L}{2}\right)} = \omega(2^{10 \log n}),$$

and the lemma follows since $X_j \leq \hat{X}$. \square

Here is how the results on the upper and the lower part fit together.

Proposition 10.8. For all Boolean functions f ,

$$\Psi(f) \geq W - W^b - X_{\Delta}(f).$$

Proof. Let G be a minimal FBDD for f . Then by definition,

$$\Psi(f) = \text{size}(G) = \sum_{j=1}^n \text{level}(G, j).$$

Using Lemma 10.5, we see that

$$\Psi(f) \geq \sum_{j \in [i_0]} k_j - X_{\Delta, j}(f) = W - W^b - X_{\Delta}(f). \quad \square$$

10.3 The Minimal FBDD Size of Random Boolean Functions

Now we put things together and prove Main Theorem 2 (i). The expected size of minimal FBDDs is given by the next theorem.

Theorem 10.9. If $n \rightarrow +\infty$ is such that $a(n) \notin [-\log^2 n - \omega(1) .. \omega(1)]$, then

$$\mathbf{E}(\Psi) \sim W.$$

Moreover, $|a| \geq \sqrt{n}$ implies

$$\mathbf{E}(\Psi) = \left(1 - 2^{-\Omega(\sqrt{n})}\right) W.$$

Proof. We split the sequence of n into two subsequences according to the sign of a .

The case $a = \omega(1)$:

Since $i' > i_0$, we have $\hat{X} \leq X - X_{i'}$, and by Theorem 3.8 (iii),

$$\mathbf{E}(\hat{X}) \leq \mathbf{E}(X - X_{i'}) = 2^{L - \Omega(\sqrt{n})}.$$

So Lemma 10.7 implies

$$\mathbf{E}(X_{\Delta}) = o\left(2^{\log^2 n} \cdot 2^{L - \Omega(\sqrt{n})} + 1\right) = 2^{L - \Omega(\sqrt{n})} = o(1).$$

Also $W^b = o(W)$ by Lemma 10.6. For $a > \sqrt{n}$ we even have $W^b/W = O(2^{L-\sqrt{n}})$. So by Proposition 10.8, $E(\Psi) \sim W$, and $a > \sqrt{n}$ implies $E(\Psi) = (1 - 2^{-\Omega(\sqrt{n})})W$.

The case $a = -\log^2 n - \omega(1)$:

Here we use the upper bound $\hat{X} \leq X$. By Theorem 3.8 (iii),

$$E(\hat{X}) = E(X_{i'}) + 2^{L-\Omega(\sqrt{n})}.$$

Using Theorem 3.8 (i) and Proposition 3.12, $E(X_{i'}) \sim 2^{L+a-1}$. For $a < -\sqrt{n}$, we have $E(X_{i'}) = 2^{L-\Omega(\sqrt{n})}$ by Lemma 3.9. Therefore, Lemma 10.7 implies that

$$E(X_\Delta) = o\left(2^{\log^2 n} \cdot 2^{L+a-1} + 1\right) = o\left(2^{L-\omega(1)} + 1\right) = o(W),$$

and a similar calculation shows that $E(X_\Delta) = 2^{L-\Omega(\sqrt{n})}$ for $a < -\sqrt{n}$. The relative size of W^b is only $W^b/W = O(2^{L/\sqrt{2}})$ by Lemma 10.6. \square

Perhaps the following brief statement gives a better understanding why the $\log^2 n$ in Theorem 10.9 cannot be improved easily: If $a = -\omega(1)$ and $a = o(\sqrt{n})$, then we have $E(X_{i'})/2^L \sim 2^{a-1}$ by Proposition 3.12, and this has to be multiplied by

$$\#\Delta(i') = \binom{n}{\log n + O(1)} \geq 2^{(\log n + O(1))(\log n - \log \log n + o(1))}.$$

So either $a = -\Omega(\log^2 n)$ or a better idea for a proof is necessary.

Finally, we show that Theorem 10.9 does not only hold for the expected minimal FBDD size $E(\Psi)$, but also for Ψ itself with high probability.

Theorem 10.10. If $n \rightarrow +\infty$ is such that $a(n) \notin [-\log^2 n - \omega(1) .. \omega(1)]$, then

$$\Pr\left(\Psi \leq (1 - o(1))W\right) = e^{-\Omega(2^{L/\log n})}.$$

Moreover, if $|a| > \sqrt{n}$, then for all $\lambda \geq 1$,

$$\Pr\left(\Psi \leq \left(1 - 2^{-\Omega(\sqrt{n})}\right)W\right) = e^{-\Omega(2^{L/\log n})}.$$

Proof. The heart of the proof of Theorem 10.9 was the asymptotic estimation of $E(X_\Delta)$ and W^b and an application of Proposition 10.8. So we are done if we can show that $X_\Delta - E(X_\Delta)$ is small with high probability.

Actually, we have to use a refinement of Proposition 10.8. Let

$$\tilde{X}_{\Delta,j}(f) := \min \{k_j, X_{\Delta,j}\} = \min \left\{ k_j, \sum_{d \in \Delta(j)} X_{\pi_d,j}(f) \right\}.$$

Since $\text{level}(G, j)$ is nonnegative, we can replace $X_{\Delta,j}$ with $\tilde{X}_{\Delta,j}$ in Lemma 10.5. Similarly, in Lemma 10.7 we can replace X_{Δ} with $\tilde{X}_{\Delta} := \sum_{j \in [i_0]} \tilde{X}_{\Delta,j} \leq X_{\Delta}$. Here $\tilde{X}_{\Delta} \leq k_{\tilde{i}} + \tilde{X}_{\Delta}^b$, where $\tilde{X}_{\Delta}^b := \sum_{j=[\tilde{i}]}^{\lfloor i_0 \rfloor} \tilde{X}_{\Delta,j}$ and \tilde{i} will be specified below. Then in Proposition 10.8, we have

$$\Psi(f) \geq W - W^b - k_{\tilde{i}} - \tilde{X}_{\Delta}^b. \quad (10.2)$$

Observe that $\tilde{X}_{\Delta}^b(f)$ is a sum over $\leq n^{n-\tilde{i}+10}$ summands of the form $X_{\pi_d,j}(f)$, $d \in \Delta(j)$, and each $X_{\pi_d,j}$ is no more than $2^{\frac{1+c}{2}L}$ away from its expected value with probability $1 - 2e^{-2^{cL/4}}$ by (4.3) from the proof of Theorem 4.3, for each $c > 0$. We set $\tilde{i} := n - n^{2/3}$, so that $k_{\tilde{i}} = O(2^{n-n^{2/3}}) = W \cdot 2^{-\Omega(\sqrt{n})}$. Also, for $c := 1/\log n$, we have $n^{n-\tilde{i}+10} \cdot 2^{\frac{1+c}{2}L} = 2^{(\frac{1}{2}+O(1/\log n))L}$, so that

$$\Pr \left(|\tilde{X}_{\Delta}^b - \mathbb{E}(\tilde{X}_{\Delta}^b)| \geq 2^{(\frac{1}{2}+O(1/\log n))L} \right) \leq 2^{O(n^{2/3})} e^{-2^{L/\log n/4}}.$$

Since $\tilde{X}_{\Delta}^b \leq X_{\Delta}$, the estimations on the lower bound from Proposition 10.8 in the proof of Theorem 10.9 also apply to the improved lower bound (10.2) up to $k_{\tilde{i}} = 2^{-\Omega(\sqrt{n})} \cdot W$. Therefore, the asymptotics for $\mathbb{E}(\Psi)$ from the proof of Theorem 10.9 are valid for Ψ with probability $e^{-\Omega(2^{L/\log n})}$, too. \square

Thus, Main Theorem 2 (i) is proved.

10.4 Some Remarks on Main Theorem 2 (i)

We end this section with some ideas how Theorems 10.9 and 10.10 might be improved.

- (1) The cut-off point in Theorem 10.10 can be pushed closer to W in some cases. E. g., for $a \leq 0$, W^b is much smaller than just $o(W)$, but still we cannot achieve a lower bound of $\mathbb{E}(\Psi) - 2n2^{\frac{1+c}{2}L}$ as for OBDDs, see Corollary 5.6.
- (2) Use a better bound than W^b for the lower part.
- (3) Show that we do not really have to sum up the reduction effect for different variable orderings, due to incompatibilities among them.
- (4) It is fairly easy to come up with results for a *fixed* graph ordering. (But the number of graph orderings is so large that even our doubly exponential probability bounds are not strong enough to make the simple proof technique from the OBDD analysis carry over.)

- (5) So one should try to reduce the number of graph orderings which have to be considered, or exploit dependencies among variables which correspond to ‘similar’ graph orderings. (We worked on this for a while, but it seems to become *very* technical.)

We believe that the \log^2 in Theorem 10.10 is not best possible. More refined methods seem to be necessary to close the gap between Assertions (i) and (ii) of Main Theorem 2.

11 Small FBDDs and large OBDDs

This section contains the proof of Main Theorem 2 (ii). We show that there exists a certain range of sequences of n such that for a random Boolean function with n variables, the minimal OBDD size is $Z_* = (1 + o(1)) W'(n)$ and the minimal FBDD size is $\Psi = (1 - \Omega(1)) W'(n)$, with high probability. Thus, FBDDs are a constant factor smaller than OBDDs. The upper bound follows from the probabilistic analysis of the performance of the algorithm `SIMPLETYPE`.

11.1 Algorithm `SIMPLETYPE`

The idea behind algorithm `SIMPLETYPE` is that minimal FBDDs should be smaller than minimal OBDDs because they can somehow ‘collect’ mergings which are possible within OBDDs for different variable orderings. While a lot of ideas come into one’s mind when thinking about a way how the additional reduction potential of FBDDs that arises from the absence of a global variable ordering could be exploited algorithmically, it turns out to be rather difficult to come up with an algorithm that has a *provable* performance guarantee. Note that we are not interested in (possibly rare) worst-case inputs; `SIMPLETYPE` is a *deterministic algorithm* which performs ‘well’ on *random inputs*.

In the application of FBDDs, reasonable FBDD types are found using heuristics which exploit, for example, circuit structure [BMS96]. Of course, such an approach is not feasible in our context. Let us emphasise that Algorithm `SIMPLETYPE` is *not* meant to be used in practice, but serves us to derive an *existence* result in a *constructive* way. Consequently, we shall give no complexity analysis here, but it is clear that the running time is not ‘pathological’ (i. e., primitive recursive).

Algorithm `SIMPLETYPE` has three phases. In the first phase, we perform Shannon decomposition up to some level i^\sharp , using the canonical variable ordering (any other variable ordering does equally well, due to the symmetry of the probability space). Recall that the analysis of random OBDDs showed that the ‘gap’ between expected and worst-case size is almost entirely concentrated on the ‘critical’ level i' , if there is such a gap at all. Later in the probabilistic analysis, we will consider sequences of n such that the OBDD gap is rather small – less than a constant fraction of W – but not extremely small. The parameters can be adjusted in such a way that for some constant fraction of the nodes at level i^\sharp which were constructed in Phase 1, there is a possible merging *at level i'* below each node for *some* variable ordering which coincides with the canonical one on the first i^\sharp levels. Also, i^\sharp is not very far above i' , so that the size of level i^\sharp is a constant fraction of W . For technical reasons, we will only consider those nodes at level i^\sharp which have a *unique* ‘partner’. These are called ‘uniquely partly mergeable’. Now in Phase 2, we choose an appropriate decomposition ordering below the uniquely partly mergeable nodes at level i^\sharp and use the canonical one otherwise to obtain a binary decision tree. Finally, we apply the merging rule in Phase 3.

Algorithm SIMPLETYPE

Input: A Boolean function $f = f(x_1, \dots, x_n)$ and $i^\sharp, i^\flat \in [n]$, $i^\sharp < i' < i^\flat$.

Output: A quasireduced FBDD G representing f .

Phase 1:

- At the first $i^\sharp - 1$ levels, we perform Shannon decomposition with respect to the canonical variable ordering. This determines the subfunctions to be represented at level i^\sharp .

Phase 2:

- To describe the order in which the remaining variables are tested, we use the following technical notions:

- Let $\tilde{\Xi} := \{d \subseteq \{x_j \mid j \in [i^\sharp .. i^\flat - 1]\} \mid \#d = i' - i^\sharp\}$ and denote for $d \in \tilde{\Xi}$ the partial assignment which sets the variables in d to 0 by \bar{d} .
- Define a relation of being ‘partly mergeable²’ on $\text{Level}(G, i^\sharp)$ by

$$M(v, v') \quad :\leftrightarrow \quad v \neq v' \wedge \exists d \in \tilde{\Xi} : F(v)_{\bar{d}} = F(v')_{\bar{d}}$$

and denote the set of ‘uniquely partly mergeable³’ nodes by

$$U := \{v \mid \exists^1 v^* : M(v, v^*)\}.$$

- For each partly mergeable pair $v, v^* \in U$, choose *one* $d(v) = d(v^*) \in \tilde{\Xi}$ arbitrarily among those which satisfy $F(v)_{\overline{d(v)}} = F(v^*)_{\overline{d(v)}}$.
- For the computation paths going through some uniquely partly mergeable $v \in U$, continue Shannon decomposition according to the variable ordering $\pi_{\{x_1, \dots, x_{i^\sharp-1}\} \cup d(v)}$. For the remaining $v \in \text{Level}(G, i^\sharp) \setminus U$, use the canonical variable ordering.

Phase 3:

- Apply the merging rule to get the qFBDD G .

SIMPLETYPE makes no use of the deletion rule at all, but in view of Corollary 9.5 and Theorem 9.6, only a marginal improvement would be possible. Note however, that the estimations on the effect of the deletion rule were proved for strongly reduced FBDDs. It seems hard to show that one will get a strongly reduced FBDD by applying the deletion rule to the result of SIMPLETYPE (and we conjecture that this is even false in general).

² In german: ‘teilverschmelzbar’

³ ‘eindeutig teilverschmelzbar’

The parameter i^b will be used for fine-tuning of the size of $\check{\Xi}$ ensuring that the set U is large. For too small $\#U$, there are only a few partly mergings. For too large $\#U$, not enough partly mergings satisfy the uniqueness condition.

The restriction to uniquely partly mergeable nodes is made for technical reasons, but we do not expect the number of partly mergeable nodes to be much (i. e., more than a *constant* factor) larger, at least in the interesting cases. See the remarks on “how *not* to improve algorithm SIMPLETYPE” at the end of this section.

11.2 The Expected Size of the Set U

By choosing the right decomposition variables at and below level i^\sharp , SIMPLETYPE can save some nodes at level i' which lie below the uniquely partly mergeable nodes at level i^\sharp in the topological ordering. This is why we use the term “partly mergeable”.

Proposition 11.1. Let $G := \text{SIMPLETYPE}(f, i^\sharp, i^b)$. Then

$$\text{level}(G, i') \leq k_{i'} - \frac{\#U}{2}.$$

Proof. For each pair $v, v' \in U$ such that $M(v, v')$, we can perform a merging among the successor nodes representing $F(v)_{\bar{d}(v)}$ and $F(v')_{\bar{d}(v')}$ at level i' . \square

Hence, we must show that $\#U$ is large. The first step to estimate the size of U is to determine the probability that two nodes at level i^\sharp are partly mergeable (not necessarily uniquely).

Lemma 11.2. Let $G := \text{SIMPLETYPE}(f, i^\sharp, i^b)$. Then for any two nodes $v, v' \in \text{Level}(G, i^\sharp)$,

$$\Pr(M(v, v')) = \frac{\#\check{\Xi}}{m_{i'}} \left(1 - \left| O\left(\frac{\#\check{\Xi}}{\sqrt{m_{i'}}}\right) \right| \right).$$

Proof. Let $v, v' \in \text{Level}(G, i^\sharp)$ after Phase 1 of SIMPLETYPE. The two nodes v and v' are partly mergeable if and only if there exists a $d \in \check{\Xi}$ such that $F(v)_{\bar{d}} = F(v')_{\bar{d}}$. We estimate the probability of this event using the “principle of inclusion and exclusion”.

The Bonferroni inequalities tell us that

$$\Pr(M(v, v')) \begin{cases} \leq \sum_{d \in \tilde{\Xi}} \Pr(F(v)_{\bar{d}} = F(v')_{\bar{d}}) =: p^\sharp; \\ \geq p^\sharp - \sum_{\substack{\{d, d'\} \subseteq \tilde{\Xi} \\ d \neq d'}} \Pr(F(v)_{\bar{d}} = F(v')_{\bar{d}} \wedge F(v)_{\bar{d}'} = F(v')_{\bar{d}'}) =: p^\flat. \end{cases}$$

Note that $F(v)$ and $F(v')$ are independent random Boolean functions of $n - i^\sharp + 1$ variables. Let $g := F(v) \oplus F(v')$. Then $F(v)_{\bar{d}} = F(v')_{\bar{d}} \leftrightarrow g_{\bar{d}} = 0$ for all d .

To compute the probabilities, we count the number of unconstrained truth table positions of g . A position c is called *constrained* if and only if $g_{\bar{d}} = 0$ implies $g(c) = 0$. If we want to produce a g satisfying $g_{\bar{d}} = 0$, we are free to choose the value of $g(c)$ if and only if c is unconstrained. Thus, each constrained position ‘halves’ the probability of the event $g_{\bar{d}} = 0$. The number of constrained truth table positions can itself be counted using the probabilistic method. Think of c as a random variable taking uniformly distributed values in $^{[n-i^\sharp+1]}2$. Then

$$\begin{aligned} \log \Pr_g(g_{\bar{d}} = 0) &= -\#\{c \mid c \text{ is constrained}\} \\ &= -\#\{c \mid g_{\bar{d}} = 0 \text{ forces } g(c) = 0\} \\ &= -2^{n-i^\sharp+1} \Pr_c(g_{\bar{d}} = 0 \text{ forces } g(c) = 0) \\ &= -2^{n-i^\sharp+1} \Pr_c(\forall x_i \in d: c_i = 0) \\ &= -2^{n-i^\sharp+1} 2^{-\#d} \\ &= -2^{n-i'+1}. \end{aligned}$$

Similarly,

$$\begin{aligned} \log \Pr_g(g_{\bar{d}} = 0 \wedge g_{\bar{d}'} = 0) &= -2^{n-i^\sharp+1} \Pr_c(\forall x_i \in d: c_i = 0 \vee \forall x_i \in d': c_i = 0) \\ &= -2^{n-i^\sharp+1} \left(2 \Pr_c(\forall x_i \in d: c_i = 0) - \Pr_c(\forall x_i \in d \cup d': c_i = 0) \right) \\ &= -2^{n-i^\sharp+1} (2 \cdot 2^{-\#d} - 2^{-\#d \cup d'}) \\ &= -2^{n-i^\sharp-\#d+2} + 2^{n-i^\sharp-\#d \cup d'+1} \\ &= -2^{n-i'+2} + 2^{n-i'-\#d \setminus d'+1} \\ &\leq -\frac{3}{2} \cdot 2^{n-i'+1}. \end{aligned}$$

Therefore,

$$p^\sharp = \#\check{\xi} \cdot 2^{-2^{n-i'+1}} = \frac{\#\check{\xi}}{m_{i'}}$$

and

$$p^\flat \geq p^\sharp - \#\check{\xi}^2 \cdot 2^{-\frac{3}{2}2^{n-i'+1}} = p^\sharp \left(1 - \frac{\#\check{\xi}}{\sqrt{m_{i'}}}\right). \quad \square$$

Next we compute the probability that a node is *uniquely* partly mergeable and the expected size of U .

Lemma 11.3. Let $G := \text{SIMPLETYPE}(f, i^\sharp, i^\flat)$ and $k := k_{i^\sharp}$. For two nodes $v, v' \in \text{Level}(G, i^\sharp)$, let

$$p := \Pr(M(v, v')) \quad \text{and} \quad p' := \Pr(\exists^{-1}v^* : M(v, v^*)).$$

Then $\mathbb{E}(\#U) = kp'$, and

$$p' = (k-1)p(1-p)^{k-2} = \begin{cases} o(1), & p = o(1/k); \\ \Theta(1), & p = \Theta(1/k); \\ o(1), & p = \omega(1/k). \end{cases}$$

Proof. The equation $\mathbb{E}(\#U) = kp'$ is just the linearity of expectations.

For each v , $\#\{v^* \mid M(v, v^*)\}$ is a sum of independent identically distributed indicator random variables and therefore binomially distributed with parameters $k-1$ and p . So

$$p' = \binom{k-1}{1} p^1 (1-p)^{(k-1)-1} = (k-1)p(1-p)^{k-2}.$$

Finally, the asymptotics hold since

$$(1-p)^{k-2} = \left(1 - \frac{p(k-2)}{k-2}\right)^{k-2} \begin{cases} \leq e^{-p(k-2)} \leq 1, & \text{for all } p, \text{ and} \\ \sim e^{-pk} & \text{for } p = \Theta(1/k). \end{cases} \quad \square$$

If n is in a certain range, then by the following theorem we can arrange the settings of i^\sharp and i^\flat for SIMPLETYPE such that the expected size of $\#U$ for random f is at least some constant factor of w_{i^\sharp} . As a consequence, level i' is not completely ‘full’.

Theorem 11.4. If $n \rightarrow +\infty$ in such a way that $0 > a(n) = O(\log \log n)$, then there exist parameters $i^\#$ and i^b for SIMPLETYPE so that $\mathbb{E}(\#U) = \Omega(2^{L(n)})$.

Proof. We have $k_{i'} = \Omega(2^L)$. Assume that we can find $i^\#$ and i^b such that

$$\#\check{\Xi} = \Theta\left(\frac{m_{i'}}{k_{i'}}\right) \quad \text{and} \quad i' - i^\# = O(1). \quad (11.1)$$

Then in Lemma 11.2, we have

$$\frac{\#\check{\Xi}}{\sqrt{m_{i'}}} = O\left(\frac{\sqrt{m_{i'}}}{k_{i'}}\right) = O\left(2^{2^{o(1)-1}L+O(1)-L}\right) = o(1)$$

and hence, the p in Lemma 11.3 is

$$p \sim \frac{\#\check{\Xi}}{m_{i'}} = \Theta\left(\frac{1}{k_{i'}}\right) = \Theta\left(\frac{1}{k_{i^\#}}\right),$$

which implies $p' = \Theta(1)$ and finally,

$$\mathbb{E}(\#U) = k_{i^\#} p' = \Omega(k_{i'} p') = \Omega(2^L).$$

So let us see how to satisfy (11.1). We have

$$\#\check{\Xi} = \binom{i^b - i^\#}{i' - i^\#} = \binom{b + c}{b},$$

writing $b := i' - i^\#$ and $c := i^b - i'$. It turns out that $b := \lceil \frac{-2a}{\log \log n} \rceil$ is a good choice. Observe that $1 \leq b = O(1)$ as required. On the other hand, by (3.11) and Lemma 3.7,

$$\frac{m_{i'}}{k_{i'}} = 2^{(2^{-\delta'} - 1)L - \delta'} = 2^{-a(1+o(1))}.$$

So we choose c such that

$$\log \binom{b+c}{b} = (2^{-\delta'} - 1)L + O(1),$$

minimising the $O(1)$ term. (This will imply (11.1).) Since

$$\binom{b+c+1}{b} / \binom{b+c}{b} = \frac{b+c+1}{c+1} = O(1),$$

such a c clearly exists. But can it happen that we need $c > n - i' = \log n + O(1)$ sometimes? – No, because $\log \binom{b+c}{b}$ can become as large as

$$\log \binom{b + \log n + O(1)}{b} = b \log(b + \log n + O(1)) + O(1) = b \log \log n + O(1)$$

and we only need to achieve a value of

$$(2^{-\delta'} - 1)L \sim -a \leq \frac{b \log \log n}{2}. \quad \square$$

11.3 Large Deviations from the Expected Size of the Set U

To show that $\#U$ is not much smaller than $\mathbb{E}(\#U)$ with high probability, we need a large deviation inequality. We will apply Azuma's martingale inequality once more. The first step is to check that $\#U$ indeed satisfies a (somewhat relaxed) Lipschitz condition.

Lemma 11.5. In the framework of algorithm SIMPLETYPE, let f and f' be Boolean functions such that

$$\#\{\alpha \mid \alpha \in \tilde{\Xi} \wedge \overline{\text{free}}(\alpha) = \{x_j \mid j < i^\#\} \wedge f_\alpha \neq f'_\alpha\} \leq 1.$$

Then $|\#U(f) - \#U(f')| \leq 2(\#\tilde{\Xi} + 1)$.

Proof. There is nothing to prove if $f = f'$. So assume that α is the (unique) partial assignment with $\overline{\text{free}}(\alpha) = \{x_j \mid j < i^\#\}$ such that $f_\alpha \neq f'_\alpha$. Let v be the corresponding node at level $i^\#$ after Phase 1, whose subfunction changed from f_α to f'_α .

The relation M defines a graph structure on $\text{Level}(G, i^\#)$. An edge between w and w' is labelled with $\{d \in \tilde{\Xi} \mid F(w)_{\bar{d}} = F(w')_{\bar{d}}\}$, which is nonempty. Only edges incident to v can change if we replace f_α by f'_α . We split this replacement into two steps.

In the first step, we remove all edges incident to v . If $v \in U(f)$, this takes two nodes away from U . On the other hand, new uniquely partly mergeable pairs can originate, if at least one of their nodes was formerly adjacent to v . The number of such pairs is bounded by $\#\tilde{\Xi}$, because the set labels are disjoint for removed edges corresponding to different pairs. So $\#U$ can increase by at most $2\#\tilde{\Xi}$.

In the second step, we insert the new edges incident to v according to f' . By symmetry, $\#U$ can increase by at most 2 and decrease by at most $2\#\tilde{\Xi}$. It follows that $|\#U(f) - \#U(f')| \leq 2(\#\tilde{\Xi} + 1)$. \square

Now the proof of the large deviation result is straightforward.

Theorem 11.6. Under the assumptions and with the parameter settings of Theorem 11.4,

$$\Pr \left(|\#U - \mathbb{E}(\#U)| \geq \frac{\mathbb{E}(\#U)}{2} \right) \leq e^{-\Omega(2^L / (\log \log n)^2)}.$$

Proof. By Lemma 11.5, $\frac{\#U(\cdot)}{2(\#\check{\Xi}+1)}$ satisfies the Lipschitz condition (4.1). Therefore by Azuma's inequality (Theorem 4.1),

$$\Pr \left(|\#U - \mathbb{E}(\#U)| \geq \lambda \cdot 2(\#\check{\Xi} + 1) \cdot \sqrt{k_{i\ddagger}} \right) \leq 2e^{-\lambda^2/2}.$$

Note that $\#\check{\Xi} = \binom{O(\log n)}{O(1)} = (\log n)^{O(1)}$, so $\log \#\check{\Xi} = O(\log \log n)$. Setting

$$\lambda := \frac{\mathbb{E}(\#U)}{4(\#\check{\Xi} + 1)\sqrt{k_{i\ddagger}}} = \Omega(2^{L/2}/\log \log n),$$

we get

$$\Pr \left(|\#U - \mathbb{E}(\#U)| \geq \frac{\mathbb{E}(\#U)}{2} \right) \leq 2e^{-\Omega(2^L/(\log \log n)^2)}. \quad \square$$

11.4 Small FBDDs via SIMPLETYPE

Our probabilistic analysis of SIMPLETYPE can be summarised as follows.

Theorem 11.7. Let $n \rightarrow +\infty$ in such a way that $0 > a(n) = O(\log \log n)$. Then with the parameter settings from Theorem 11.4,

$$\Pr_f \left(\text{size}(\text{SIMPLETYPE}(f)) \geq (1 - \Omega(1))W(n) \right) = e^{-\Omega(2^L/(\log \log n)^2)}.$$

Thus, FBDDs are a constant factor 'better' than OBDDs on average for sequences of n such that $a(n) = \omega(1) \wedge a(n) = O(\log \log n)$.

Proof. Let $G := \text{SIMPLETYPE}(f)$. For $j \in [i' - 1]$, we apply the worst-case bound $\text{level}(G, j) \leq k_j$.

By Theorems 11.4 and 11.6,

$$\Pr(\#U \leq \Omega(2^n)) = e^{-\Omega(2^L/(\log \log n)^2)},$$

so using Proposition 11.1,

$$\Pr(\text{level}(G, i') \geq k_{i'} - \Omega(2^n)) = e^{-\Omega(2^L/(\log \log n)^2)}.$$

This gives us a constant factor 'gap' at level i' .

However, the width of the levels $j \in [i' + 1 .. n]$ is no longer bounded by m_j in the FBDD case. For example, level n might well contain $2n$ nodes, testing all the

variables. But we can multiply m_j by $\#\check{\Xi}$, the number of variable orderings that can possibly occur on any computation path in G , to get an upper bound. Fortunately, $\#\check{\Xi} \leq 2^{n-i^\#} = 2^{O(\log n)} = n^{O(1)}$ is a rather small number. Using (2.7), we get

$$\sum_{j=i'+1}^n \text{level}(G, j) \leq n^{O(1)} O\left(2^{2^n - \lceil L \rceil}\right).$$

Since $\delta' < 0$, we have $\lceil L \rceil > L + 1/2$, so

$$n^{O(1)} 2^{2^n - \lceil L \rceil} \leq n^{O(1)} 2^{2^{\log L - 1/2}} = 2^{(1/\sqrt{2} + o(1))L} = o(W),$$

which completes the proof of the theorem. \square

Thus, Main Theorem 2 (ii) has been proved.

11.5 How Not to Improve Algorithm SIMPLETYPE

We conclude with some remarks on how *not* to improve Theorem 11.7, and speculate about what else might be provable.

The challenge in using the additional power of FBDDs to get a better upper bound on the optimal size is to arrange the process of Shannon decomposition in such a way that the achieved reduction (which is always mostly due to the merging rule) is as large as possible.

In the framework of algorithm SIMPLETYPE, there is a close connection between p' (or $\#U$) and $\#\check{\Xi}$. In order to keep $p' = \Theta(1/k_{i^\#})$ (see Lemma 11.3), $\#\check{\Xi}$ must become larger as a approaches $-\infty$. This is mostly controlled by the value of $i' - i^\#$, which is called b in the proof of Theorem 11.4. But since at most one merging is counted below each node at level $i^\#$, we are not allowed to let $b \rightarrow +\infty$. We must keep $b = O(1)$ in order to get $\#U = \Omega(2^L)$.

So the question arises if we can enlarge p by other means. One idea is to relax the relation of being partly mergeable. We could use

$$\begin{aligned} \tilde{M}(v, v') &: \leftrightarrow v \neq v' \wedge \exists d \subseteq \{x_j \mid j \geq i^\#\}: \\ &(\#d = i' - i^\# \wedge \exists \alpha, \alpha': \overline{\text{free}}(\alpha) = \overline{\text{free}}(\alpha') = d \wedge F(v)_\alpha = F(v')_{\alpha'}) \end{aligned}$$

instead of $M(v, v')$. In this way, we might hope to increase p by a factor of 2^{2^b} . But this is just $O(1)$, as long as $b = O(1)$.

Another idea would be not to neglect nodes which are partly mergeable, but not uniquely partly mergeable. The restriction to uniquely partly mergeable nodes was made to simplify the analysis. If a node can participate in several ‘partly mergings’, conflicts can arise between the corresponding variable orderings, if they are different. It is not always possible to ‘realize’ such mergings. However, to estimate the power of this

approach, assume contrafactually for a moment that no conflicts would arise. Then we still have the problem that p decreases as $-a$ increases. By the proof of Theorem 11.3, the number of partly mergings in which a node is involved has a binomial distribution with parameters $k_{i\#} - 1$ and p . Therefore, for $p = o(1/k_{i\#})$, almost all partly mergings are unique. So this approach will not get us beyond $a = O(\log \log n)$, too. —

Of course, the size of the ‘Shannon gap’ in Theorem 11.7 could be determined more precisely. Also, one might try to improve SIMPLETYPE (or analyse it better) such that the upper bound matches the lower bound for general BDDs for some parametrisation of n for which there is really ‘room’ for such an improvement. Such a result would be very interesting and surprising, because it implies that FBDDs alternate between BDD-like and OBDD-like behaviour on random Boolean Functions for different parametrisations of n . Stated in another way, the question is whether the read-once property or the variable ordering restriction is the larger step from general BDDs to OBDDs. Although we have not proved this, it seems fairly natural from Main Theorem 2 that parametrisations do exist for which all the three sizes are separated by constant factors.

12 Appendix

12.1 Notation

O-terms. The Landau symbols O and o are used everywhere in computer science, but there seems to be little agreement in the literature with respect of the signs. The confusion seems to be even greater for Ω , see e.,g. [OW93] for a nonstandard definition. ω s are a bit unusual (though not our invention), but come in handy sometimes. So let us briefly define the way we will use them. The policy we adopt is that O and o are unsigned, whereas Ω and ω are nonnegative. We say that

$$\begin{aligned}f = O(g) & \text{ iff } \exists c > 0 \exists n_c \forall n \geq n_c: |f(n)| \leq c \cdot g(n), \\f = \Omega(g) & \text{ iff } \exists c > 0 \exists n_c \forall n \geq n_c: f(n) \geq c \cdot g(n), \\f = o(g) & \text{ iff } \forall c > 0 \exists n_c \forall n \geq n_c: |f(n)| \leq c \cdot g(n), \\f = \omega(g) & \text{ iff } \forall c > 0 \exists n_c \forall n \geq n_c: f(n) \geq c \cdot g(n), \\f = \Theta(g) & \text{ iff } f = O(g) \text{ and } f = \Omega(g), \\f \sim g & \text{ iff } f = (1 + o(1))g.\end{aligned}$$

Intervals of integers are denoted as $[a..b] := [a, b] \cap \mathbb{Z}$ and $[a] := [1..a]$.

Fractions. We write ab/cd for $\frac{ab}{cd}$.

Logarithms. Unless stated otherwise, all logarithms have base 2: $\log \equiv \log_2$. The natural logarithm is denoted by $\ln \equiv \log_e$.

12.2 Inequalities

The following inequalities are used occasionally.

For $x \in \mathbb{R}$, $e^x \geq 1 + x$ and $e^x \geq 1 + x + \frac{x^2}{2} + \frac{x^3}{6}$.

For $x < 0$, $e^x \leq 1 + x + \frac{x^2}{2}$.

For $x < 1$, $e^x \leq \frac{1}{1-x}$.

For $x > -1$, $\log(1+x) \leq x \log e$.

12.3 Asymptotics

exp. For $x = o(1)$,

$$\begin{aligned}e^x &= 1 + x + \frac{x^2}{2} + O(x^3) = 1 + x + \frac{x^2}{2}(1 + o(1)) \\ &= 1 + x + O(x^2) = 1 + x(1 + o(1)) = 1 + o(1).\end{aligned}$$

log. For $x = o(1)$,

$$\log(1+x) = 1 + x \log e + O(x^2) = x \log e(1 + o(1)) = o(1).$$

L. The function L is defined in Proposition 2.2 on page 32. For $n \rightarrow +\infty$,

$$\begin{aligned} n &= L + \log L = L + \log(n - \log L) \\ &= L + \log n + o(1) \sim L \end{aligned}$$

and

$$2^L = 2^{n/L} \sim 2^n/n.$$

δ' , \tilde{a} , i' , $k_{i'}$, $m_{i'}$. These are defined in Proposition 1.1, page 24 (k_i, m_i) and Section 3.1, page 42 ($\delta' = \delta_{\tilde{a}}$, $i' = i_{\delta'} = L + 1 + \delta'$, $m_{i'} = 2^{2^{-\delta'}L}$, $k_{i'} = 2^{\delta'+L}$). For the convenience of the reader, we repeat some frequently used estimations in the following table.

then If ...	$a = o(n)$	$a = O(\sqrt{n})$	$a = o(\sqrt{n})$	$a = O(1)$
$\tilde{a} =$	$a(1 + o(1))$	$a + O(1)$	$a + o(1)$	$O(1)$
$\delta' =$	$o(1)$	$O(1/\sqrt{n})$	$o(1/\sqrt{n})$	$O(1/n)$
$\frac{k_{i'}}{2^L} =$	$2^{o(1)} \sim 1$			
$\frac{m_{i'}}{2^L} =$	$2^{a(1+o(1))}$	$2^{a+O(1)} = \Theta(2^a)$	$2^{a+o(1)} \sim 2^a$	$2^{O(1)} = \Theta(1)$

13 References

A commented list of publications of the author is given at the end of this section.

- [Ake78] Sheldon B. Akers: Binary Decision Diagrams; IEEE Transactions on Computers, vol. C-27, no. 6, 509 – 516, 1978.
- [Aro98] Sanjeev Arora: Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and Other Geometric Problems; to appear in: Journal of the ACM. (This article is based upon work presented at FOCS 1996 and FOCS 1997.)
- [AS91] Noga Alon, Joel H. Spencer: The Probabilistic Method; John Wiley & Sons, New York, 1991.
- [Bar89] D. A. Barrington: Bounded-Width Polynomial-Size Branching Programs Recognize Exactly those Languages in NC^1 ; Journal of Computer and System Sciences, vol. 38, 150 – 164, 1989.
- [BBEL96] I. Beer, S. Ben-David, C. Eisner, A. Landver: RuleBase: An Industry-Oriented Formal Verification Tool; Proceedings of the 33rd ACM/IEEE Design Automation Conference, 655 – 660, 1996.
- [BC94] Randal E. Bryant, Yirng-An Chen: Verification of Arithmetic Functions with Binary Moment Diagrams; Technical Report, Carnegie Mellon University, Pittsburgh, 37 pages, 1994.
- [BC95] Randal E. Bryant, Yirng-An Chen: Verification of Arithmetic Functions with Binary Moment Diagrams; Proceedings of the 32th ACM/IEEE Design Automation Conference, 535 – 541, 1995. See also [BC94] for more details.
- [BCL+94] Jerry R. Burch, Edmund M. Clarke, David E. Long, Kenneth L. McMillan, David L. Dill: Symbolic Model Checking for Sequential Circuit Verification; IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 13, no. 4, 401 – 424, 1994.
- [BCM+92] Jerry R. Burch, Edmund M. Clarke, Kenneth L. McMillan, David L. Dill, L. J. Hwang: Symbolic Model Checking: 10^{20} States and Beyond; Information and Computation, vol. 98, 142 – 170, 1992.
- [BCW80] M. Blum, A.K. Chandra, M.N. Wegman: Equivalence of Free Boolean Graphs can be Decided Probabilistically in Polynomial Time; Information Processing Letters, vol. 10, no. 2, 80 – 82, 1980.
- [BDT95] Bernd Becker, Rolf Drechsler, Michael Theobald: OKFDDs versus OBDDs and OFDDs; International Colloquium on Automata, Languages and Programming, Springer LNCS 944, 475 – 486, 1995.
- [BDW95] Bernd Becker, Rolf Drechsler, Ralph Werchner: On the Relation Between BDDs and FDDs; Information and Computation, vol. 123, no. 2, 185 – 197, 1995.

- [BFG+93] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, F. Somenzi: Algebraic Decision Diagrams and Their Application; Proceedings of the IEEE International Conference on Computer-Aided Design, 188 – 191, 1993.
- [BGP+97] M. Block, C. Gröpl, H. Preuß, H. J. Prömel, A. Srivastav: Efficient Ordering of State Variables and Transition Relation Partitions in Symbolic Model Checking; Technical Report, Humboldt-Universität zu Berlin, 1997.
- [BHMS84] Robert K. Brayton, Gary D. Hachtel, Curtis T. McMullen, Alberto L. Sangiovanni-Vincentelli: Logic Minimization Algorithms for VLSI Synthesis; Kluwer Academic Publishers, Boston, 1984.
- [BHR95] Y. Breitbart, H. Hunt III, D. Rosenkrantz: On the Size of Binary Decision Diagrams Representing Boolean Functions; Theoretical Computer Science, vol. 145, 45 – 69, 1995.
- [BLPV95] J. Bormann, J. Lohse, M. Payer, G. Venzl: Model Checking in Industrial Hardware Design; Proceedings of the 32nd ACM/IEEE Design Automation Conference, 298 – 303, 1995.
- [BLSW96] Beate Bollig, Martin Löbbing, Martin Sauerhoff, Ingo Wegener Complexity Theoretical Aspects of OFDDs; in: Representations of Discrete Functions, T. Sasao and M. Fujita (eds.), Kluwer Academic Publishers, 1996.
- [Blu84] N. Blum: A Boolean Function Requiring $3n$ Network Size; Theoretical Computer Science, vol. 28, 337 – 345, 1984.
- [BLW96] Beate Bollig, Martin Löbbing, Ingo Wegener: On the Effect of Local Changes in the Variable Ordering of Ordered Decision Diagrams; Information Processing Letters, vol. 59, 233 – 239, 1996.
- [BMS96] Jochen Bern, Christoph Meinel, Anna Slobodová: Some Heuristics for Generating Tree-Like FBDD Types; IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 15, no. 1, 127 – 130, 1996.
- [Bor82] K. H. Borgwardt: Some Distribution Independent Results About the Asymptotic Order of the Average Number of Pivot Steps in the Simplex Algorithm; Mathematics of Operations Research, vol. 7, 441 – 462, 1982.
- [BRB90] K. S. Brace, R. L. Rudell, R. E. Bryant: Efficient Implementation of a BDD Package; Proceedings of the 27th ACM/IEEE Design Automation Conference, 40 – 45, 1990.
- [BRSW87] R. K. Brayton, R. Rudell, A. L. Sangiovanni-Vincentelli, A. R. Wang: MIS: A Multiple-Level Interactive Logic Optimization System; IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 6, 1062 – 1081, 1987.
- [Bry86] Randal E. Bryant: Graph-Based Algorithms for Boolean Function Manipulation; IEEE Transactions on Computers, vol. C-35, no. 8, 677 – 691, 1986.

- [Bry91] Randal E. Bryant: On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Application to Integer Multiplication; *IEEE Transactions on Computers*, vol. C-40, 205 – 213, 1991.
- [Bry92] Randal E. Bryant: Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams; *ACM Computing Surveys*, vol. 24, no. 3, 293 – 318, 1992.
- [Bry95a] Randal E. Bryant: Bit-Level Analysis of an SRT Divider Circuit; Technical Report, Carnegie Mellon University, Pittsburgh, 9 pages, 1995.
- [Bry95b] Randal E. Bryant: Binary Decision Diagrams and Beyond: Enabling Technologies for Formal Verification; *Proceedings of the International Conference on Computer-Aided Design*, 1995.
- [BW96] Beate Bollig, Ingo Wegener: Improving the Variable Ordering of OBDDs is NP-Complete; *IEEE Transactions on Computers*, vol. 45, 993 – 1002, 1996.
- [CMZ+93] E. M. Clarke, K. L. McMillan, X. Zhao, M. Fujita, J. C.-Y. Yang: Spectral Transforms for Large Boolean Functions with Application to Technology Mapping; *Proceedings of the 30th ACM/IEEE Design Automation Conference*, 54 – 60, 1993.
- [Chv79] Vašek Chvátal: The Tail of the Hypergeometric Distribution; *Discrete Mathematics*, vol. 25, 285 – 287, 1979.
- [Cob66] A. Cobham: The Recognition Problem for the Set of Perfect Squares; *Proceedings of the 7. Symposium on Switching and Automata Theory*, 78 – 87, 1966.
- [Cou94] Olivier Coudert: Two-Level Logic Minimization: An Overview; *Integration*, vol. 17, 97 – 140, 1994.
- [Cou97] Olivier Coudert: Solving Graph Optimization Problems with ZBDDs; *Proceedings of the European Design & Test Conference*, 1997.
- [DB95] Rolf Drechsler, Bernd Becker: Dynamic Minimization of OKFDDs; *Proceedings of the International Conference on Computer Design*, 602 – 607, 1995.
- [DB97] Rolf Drechsler, Bernd Becker: Overview of Decision Diagrams; *IEE Proceedings Computers and Digital Techniques*, vol. 144, no. 3, 187 – 193, 1997.
- [DB98] Rolf Drechsler, Bernd Becker: *Graphenbasierte Funktionsdarstellung – Boolesche und Pseudo-Boolesche Funktionen*; Teubner Verlag, Stuttgart, 200 pages, 1998.
- [DBR96] Rolf Drechsler, Bernd Becker, Stefan Ruppertz: K^* BMDs: A New Data Structure for Verification; *European Design & Test Conference*, 2 – 8, 1996.
- [DG97] Rolf Drechsler, Nicole Göckel: Minimization of BDDs by Evolutionary Algorithms; *Proceedings of the International Workshop on Logic Synthesis, Lake Tahoe*, 1997.

- [DST+94] R. Drechsler, A. Sarabi, M. Theobald, B. Becker, M.A. Perkowski: Efficient Representation and Manipulation of Switching Functions Based on Ordered Kronecker Functional Decision Diagrams; Proceedings of the 31st ACM/IEEE Design Automation Conference, 415 – 419, 1994.
- [DTB94] Rolf Drechsler, Michael Theobald, Bernd Becker: Fast OFDD Based Minimization of Fixed Polarity Reed-Muller Expressions; Proceedings of the European Design Automation Conference, 2 – 7, 1994.
- [Ede97] Alan Edelman: The Mathematics of the Pentium Division Bug; SIAM Reviews, vol. 39, no. 1, 54 – 67, 1997.
- [Eme90] E. Allen Emerson: Temporal and Modal Logic; Chapter 16 in: Jan van Leeuwen (ed.), Handbook of Theoretical Computer Science, vol. B (Formal Models and Semantics), Elsevier Science Publishers, Amsterdam, 995 – 1072, 1990.
- [Fei96] Uriel Feige: A Threshold of $\ln n$ for Approximating Set Cover; Proceedings of the 28th Annual ACM Symposium on Theory of Computing, 314 – 318, 1996.
- [FMK88] M. Fujita, Y. Matsunaga, T. Kawato: Evaluation and Improvement of Boolean Comparison Method Based on Binary Decision Diagrams; Proceedings of the International Conference on Computer-Aided Design, 2 – 5, 1988.
- [FS90] S.J. Friedman, K.J. Supowit: Finding the Optimal Variable Ordering for Binary Decision Diagrams; IEEE Transactions on Computers, vol. 39, 710 – 713, 1990.
- [GD99] Wolfgang Günther, Rolf Drechsler: Minimization of Free BDDs; Proceedings of the Asia and South Pacific Design Automation Conference, 4 pages, 1999.
- [GPS98] Clemens Gröpl, Anand Srivastav, Hans Jürgen Prömel: Size and Structure of Random Ordered Binary Decision Diagrams (Extended Abstract); Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science, Springer LNCS 1373, 238 – 248, 1998.
- [GM94] Jordan Gergov, Christoph Meinel: Efficient Boolean Manipulation with OBDDs can be Extended to FBDDs; IEEE Transactions of Computers, vol. 43, no. 10, 1197 – 1209, 1994.
- [Hås96] Johan Håstad: Clique is Hard to Approximate Within $n^{1-\epsilon}$; Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, 627 – 636, 1996. (See ECCC TR97-038 for a newer version.)
- [Hås97] Johan Håstad: Some Optimal Inapproximability Results; Proceedings of the 29th Annual ACM Symposium on Theory of Computing, 1 – 10, 1997. (See ECCC TR97-037 for a newer version.)

- [HCO74] S. Hong, R. Cain, D. Ostapko: MINI: A Heuristic Approach for Logic Minimization; IBM Journal of Research and Development, vol. 18, 443 – 458, 1974.
- [HDB] Andreas Hett, Rolf Drechsler, Bernd Becker: The DD-Package PUMA; available via <http://www.informatik.uni-freiburg.de/FREAK/papers/puma/pumamain.html>.
- [HLJ+89] G. Hachtel, M. Lightner, R. Jacoby, C. Morrison, P. Moceyunas, D. Bostick: BOLD: The Boulder Optimal Logic Design System; in: Hawaii International Conference on System Sciences, 1989.
- [HM94] Mark A. Heap, Melvin Ray Mercer: Least Upper Bounds on OBDD Sizes; IEEE Transactions on Computers, vol. 43, no. 6, 764 – 767, 1994.
- [ISY91] N. Ishiura, H. Sawada, S. Yajima: Minimization of Binary Decision Diagrams Based on Exchanges of Variables; Proceedings of the IEEE International Conference on Computer Aided Design, 472 – 475, 1991.
- [Kar88] K. Karplus: Representing Boolean Functions with If-Then-Else DAGs; Technical Report UCSC-CRL-88-28, University of California at Santa Cruz, 1988.
- [Kar95] Michał Karoński: Random Graphs; Chapter 6 in: R. Graham, M. Grötschel, L. Lovász (eds.): Handbook of Combinatorics, vol. 1; Elsevier, Amsterdam, 1995.
- [KM72] V. Klee, G. J. Minty: How Good is the Simplex Algorithm?; In: Sisha: Inequalities - III, Academic Press, 159 – 175, 1972.
- [KMS98] D. Karger, R. Motwani, M. Sudan: Approximate Graph Coloring by Semidefinite Programming; Journal of the ACM, Vol. 45, No. 2, 246 – 265, 1998.
- [KS85] R. M. Karp, J. M. Steele: Probabilistic Analysis of Heuristics; Chapter 6 in: E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys: The Traveling Salesman Problem; Wiley, Chichester, 1985.
- [KSC78] Valentin F. Kolchin, Boris A. Sevast'ianov, Vladimir P. Chistiakov: Random Allocations; John Wiley & Sons, 1978.
- [KSR92] U. Kechschull, E. Schubert, W. Rosenstiel: Multilevel Logic Synthesis Based on Functional Decision Diagrams; Proceedings of the European Design Automation Conference, 43 – 47, 1992.
- [Lee59] C. Y. Lee: Representation of Switching Circuits by Binary Decision Programs; Bell System Technical Journal, vol. 38, 985 – 999, 1959.
- [LL90] Heh-Tyan Liaw, Chen-Shang Lin: On the OBDD-Representation of General Boolean Functions; NSC Rep., NSC79-0404-E002-35, 1990.
- [LL92] Heh-Tyan Liaw, Chen-Shang Lin: On the OBDD-Representation of General Boolean Functions; IEEE Transactions on Computers, vol. 41, no. 6, 661 – 664, 1992.

- [Lon93] David Long: Model Checking, Abstraction and Compositional Verification; Dissertation, Carnegie Mellon University, 1993.
- [LPV94] Y.-T. Lai, M. Pedram, S. B. K. Vrudhula: EVBDD-Based Algorithms for Integer Linear Programming, Spectral Transformation and Functional Decomposition; IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 13, 959 – 975, 1994.
- [LSW95] Martin Löbbing, Olaf Schröer, Ingo Wegener: The Theory of Zero-Suppressed BDDs and the Number of Knights Tours; Proceedings of the IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design, Makuhari, Chiba, Japan, 38 – 45, 1995.
- [Mas76] W. Masek: A Fast Algorithm for the String Editing Problem and Decision Graph Complexity; Master's thesis, MIT, 1976.
- [MB88] J.-C. Madre, J.-P. Billon: Proving Circuit Correctness Using Formal Comparison Between Expected and Extracted Behaviour; Proceedings of the 25th ACM/IEEE Design Automation Conference, 205 – 210, 1988.
- [McM93] Kenneth L. McMillan: Symbolic Model Checking; Kluwer Academic Publishers, Boston, 194 pages, 1993.
- [Min93] Shin-Ichi Minato: Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems; Proceedings of the 30th ACM/IEEE Design Automation Conference, 272 – 277, 1993.
- [Min96] Shin-Ichi Minato: Binary Decision diagrams and Applications for VLSI CAD; Kluwer Academic Publishers, Boston, 141 pages, 1996.
- [MIY90] S. Minato, N. Ishiura, S. Yajima: Shared Binary Decision Diagrams with Attributed Edges for Efficient Boolean Function Manipulation; Proceedings of the 27th ACM/IEEE Design Automation Conference, 52 – 57, 1990.
- [MKR92] M. R. Mercer, R. Kapur, D. E. Ross: Functional Approaches to Generating Orderings for Efficient Symbolic Representations; Proceedings of the 29th ACM/IEEE Design Automation Conference, 624 – 627, 1992.
- [MR95] Rajeev Motwani, Prabhakar Raghavan: Randomized Algorithms; Cambridge University Press, 476 pages, 1995.
- [MS97] Christoph Meinel, Anna Slobodová: Speeding up Variable Ordering of OBDDs; In: Proceedings of the International Conference on Computer Design, 1997.
- [MT98] Christoph Meinel, Thorsten Theobald: Algorithmen und Datenstrukturen im VLSI-Design; Springer Verlag, Berlin, 283 pages, 1998.
- [MWBS88] S. Malik, A. R. Wang, R. K. Brayton, A. L. Sangiovanni-Vincentelli; Logic Verification Using Binary Decision Diagrams in a Logic Synthesis Environment, Proceedings of the International Conference on Computer-Aided Design, 6 – 9, 1988.

- [OW93] Thomas Ottmann, Peter Widmayer: *Algorithmen und Datenstrukturen*; BI Wissenschaftsverlag, Mannheim, Reihe Informatik, Nr. 70, 755 pages, 1993.
- [Pap94] Christos H. Papadimitriou: *Computational Complexity*; Addison-Wesley Publishing Company, Reading, Massachusetts, 523 pages, 1994.
- [Pon95] Stephen Ponzio: A Lower Bound for Integer Multiplication with Read-Once Branching Programs; *Proceedings of the 27th Symposium on Theory of Computing*, 130 – 139, 1995.
- [PS95] S. Panda, F. Somenzi: Who are the Variables in Your Neighborhood; *Proceedings of the IEEE International Conference on Computer Aided Design*, 74 – 77, 1995.
- [PS98] Harry Preuß, Anand Srivastav: Blockwise Variable Orderings for Shared BDDs; *Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science*, Springer LNCS 1450, 1998.
- [PSP94] S. Panda, F. Somenzi, B. F. Plessier: Symmetry Detection and Dynamic Variable Ordering of Decision Diagrams; *Proceedings of the IEEE International Conference on Computer Aided Design*, 628 – 631, 1994.
- [Raz95] Ran Raz: A Parallel Repetition Theorem; *Proceedings of the 27th Annual Symposium on Theory of Computing*, 447 – 456, 1995.
- [RGB97] Rajeev K. Ranjan, Wilsin Gosti, Robert. K. Brayton, Alberto L. Sangiovanni-Vincentelli: Dynamic Reordering in a Breadth-First Manipulation Based BDD Package: Challenges and Solutions; *Proceedings of the IEEE/ACM International Conference on Computer Design*, 1997.
- [Rud93] Richard Rudell: Dynamic Variable Ordering for Ordered Binary Decision Diagrams; *Proceedings of the International Conference on Computer Aided Design*, 42 – 47, 1993.
- [Sen96] E. Sentovich: A Brief Study of BDD Package Performance; *Proceedings FMCAD*, 389 – 403, 1996.
- [Sha49] C. E. Shannon: The Synthesis of Two-Terminal Switching Circuits; *Bell System Technical Journal*, vol. 28, 59 – 98, 1949.
- [Sie94] Detlef Sieling: On the Complexity of Operations on Graph Driven BDDs and Tree Driven BDDs; *Technical Report No. 554*, Universität Dortmund, 1994.
- [Sie95] Detlef Sieling: *Algorithmen und untere Schranken für verallgemeinerte OBDDs*; Dissertation, Universität Dortmund, Shaker Verlag Aachen, 133 pages, 1995.
- [Sie98a] Detlef Sieling: On the Existence of Polynomial Time Approximation Schemes for OBDD-Minimization; *Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science*, Springer LNCS 1373, 205 – 215, 1998.

- [Sie98b] Detlef Sieling: The Nonapproximability of OBDD-Minimization; Forschungsbericht Nr. 663, Universität Dortmund, Fachbereich Informatik, 46 pages, 1998. An earlier version appeared as [Sie98a].
- [SM93] Anna Slobodová, Christoph Meinel: Efficient Manipulation with FBDDs by Means of a Modified OBDD-Package; Technical Report No. 93-09, Universität Trier, 30 pages, 1993.
- [Som96] Fabio Somenzi: CUDD: Colorado University Decision Diagram Package; available via <ftp://vlsi.colorado.edu/pub/>, 1996.
- [Spe94] Joel Spencer: Ten Lectures on the Probabilistic Method; Society for Industrial and Applied Mathematics, 88 pages, second edition 1994.
- [SRBS96] Jagesh V. Sanghavi, Rajeev K. Ranjan, Robert K. Brayton, Alberto L. Sangiovanni-Vincentelli: High Performance BDD Package Based on Exploiting Memory Hierarchy; Proceedings of ACM/IEEE Design Automation Conference, 1996. The CAL BDD-package is available via http://www-cad.eecs.berkeley.edu/Research/cal_bdd.
- [SW93] Detlef Sieling, Ingo Wegener: Reduction of OBDDs in Linear Time; Information Processing Letters, vol. 48, 139 – 144, 1993.
- [SW95] Detlef Sieling, Ingo Wegener: Graph Driven BDDs — a New Data Structure for Boolean Functions; Theoretical Computer Science, vol. 141, 283 – 310, 1995.
- [SW98a] Olaf Schröder, Ingo Wegener: The Theory of Zero-Suppressed BDDs and the Number of Knight's Tours; Formal Methods in System Design, vol. 13, 235 – 253, 1998.
- [SW98b] Detlef Sieling, Ingo Wegener: A Comparison of Free BDDs and Transformed BDDs; Technical Report No. 697, Universität Dortmund, 13 pages, 1998.
- [SWW96] Martin Sauerhoff, Ingo Wegener, Ralph Werchner: Optimal Ordered Binary Decision Diagrams for Fanout-Free Circuits; Proceedings of Synthesis and System Integration of Mixed Technologies (SASIMI), Fukuoka, Japan, 197 – 204, 1996. Submitted to Discrete Applied Mathematics.
- [TM94] C.-C. Tsai, M. Marek-Sadowska: Boolean Matching Using Generalized Reed-Muller Forms; Proceedings of the 31st ACM/IEEE Design Automation Conference, 339 – 344, 1994.
- [VIS96] The VIS Group: VIS: A System for Verification and Synthesis; Proceedings of the 8th International Conference on Computer Aided Verification, Springer LNCS 1102, 428 – 432, 1996. Homepage: <http://www-cad.eecs.berkeley.edu/Ressep/Research/vis/index.html>.
- [Weg87] Ingo Wegener: The Complexity of Boolean Functions; Teubner (Stuttgart) / Wiley (Chichester), 457 pages, 1987.

- [Weg94] Ingo Wegener: The Size of Reduced OBDDs and Optimal Read-Once Branching Programs for Almost All Boolean Functions; IEEE Transactions on Computers, vol. 43, no. 11, 1262 – 1269, 1994.

List of Publications (in reverse chronological order):

Size and Structure of Random Ordered Binary Decision Diagrams (Extended Abstract); Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science, Springer LNCS 1373, 238 – 248, 1998. (With Anand Srivastav and Hans Jürgen Prömel.)

This paper presents an earlier stage of the results from Part 1. The results from Part 2 of this dissertation are due to its author alone.

Parallel Repetition of MIP(2,1) Systems; Chapter 6 in: E. W. Mayr, H. J. Prömel, A. Steger (Eds.): Lectures on Proof Verification and Approximation Algorithms, Springer LNCS 1367, 1998. (With Martin Skutella.)

We explain the main ideas of Raz’s celebrated proof of a ‘parallel repetition theorem’ for two-prover one-round proof systems (=MIP(2,1)) [Raz95]. This is a revised version of a talk given at a research seminar for young scientists at Schloß Dagstuhl in April 1997. Raz’s parallel repetition theorem answered an important conjecture in the field of probabilistically checkable proofs to the positive and has since been applied in several important inapproximability results, including approximability thresholds for the set cover problem by Feige [Fei96] and for the clique number and other graph parameters by Håstad [Hås96,Hås97].

Efficient Ordering of State Variables and Transition Relation Partitions in Symbolic Model Checking; Technical Report, Humboldt-Universität zu Berlin, 1997. (With Mathias Block, Harry Preuß, H. J. Prömel and A. Srivastav.)

We investigate the potential of simulated annealing for the two ordering problems mentioned in the title (see also page 17). (Partitioned transition relations were introduced by the authors of [BCL+94].) These investigations were carried out within the framework of the research project ‘Effiziente Algorithmen zur formalen Verifikation von VLSI-Designs’ at Humboldt Universität zu Berlin.

Über Approximationsalgorithmen zur Färbung k -färbbarer Graphen, die vektorchromatische Zahl und andere Varianten der ϑ -Funktion; Forschungsinstitut für Diskrete Mathematik, Rheinische Friedrich-Wilhelms-Universität zu Bonn, 1996.

This diploma thesis contains investigations on the vector chromatic number, a semi-definite relaxation of the chromatic number related to the Lovász ϑ -function which has been used in the $\tilde{O}(n^{1/4})$ -colouring algorithm for 3-colourable graphs of Karger, Motwani, and Sudan [KMS98]. We show that the vectorchromatic number does not coincide with the ϑ -function, although there always exists a subgraph with the same

vectorchromatic number for which they do. Two new ‘semidefinite parameters’ are introduced and compared with the two mentioned above. The proofs are based on complementary slackness and an explicit counterexample.