The Computational Complexity Column

by

Jacobo Torán

Dept. Theoretische Informatik, Universität Ulm
Oberer Eselsberg, 89069 Ulm, Germany
jacobo.toran@uni-ulm.de

The use of computers has become very important in most areas of research, both experimental and theoretical. Curiously, in the case of complexity theory, the use of computers as a tool to guide research has not played a central role so far. Kazuyuki Amamo gives in this column an extensive overview on recent attempts to obtain new results in the area of Boolean circuits with the help of computers.

R C B
F C

Kazuyuki Amano [*]

Abstract

With the rapid advances in computers, it becomes attractive to explore the use of computers to attack open problems in computational complexity. In this article, we concentrate on the problems of the complexity of Boolean functions, and overview several recent attempts to use computers in various ways to obtain concrete results on major problems in computational complexity. We discuss the problems on several computational models including ordered binary decision diagrams, Boolean circuits, and polynomial threshold representations of Boolean functions.

[*] Tenjin 1-5-1, Kiryu, Gunma, 376-8515 Japan, Department of Computer Science, Gunma University. amano@cs.gunma-u.ac.jp

1 Introduction

Due to the rapid progress of computers, we now have a personal computer whose computational power is exceeding the power of the supercomputer two or three decades ago. My personal computer can verify (the Robertson et al.'s version [45] of) the proof of the Four Color Theorem in less than five minutes. The fields of experimental mathematics in which computation plays a central role of investigation have become increasingly wider.

In contrast, the progress of the research on computational complexity, especially on lower bound problems, is not so rapid. In spite of the computational complexity studies the nature of computation, the use of computers in the research of computational complexity seems not so common compared to mathematics. The starting point of this article is a simple thought: Can we use computers more seriously in the investigation of computational complexity?

In this article, we concentrate on the problems concerning concrete models of computations like Boolean circuits and overview several recent attempts to use computers in various ways to obtain concrete results on major problems in complexity theory. The models we consider in this article include ordered binary decision diagrams, Boolean circuits, and polynomial threshold representations of Boolean functions. We are aware that many of them are still in preliminary stages, and more work is needed to get an important result. However, we hope that these attempts inspire a new idea for attacking the major and difficult problems in complexity theory; this is one of the main points that we would like to offer in this article.

An article encouraging to use computers in the research of computational complexity, which has a similar spirit to this article, was also presented by Williams [54]. In that article, he reviewed several topics that practical computing has made a noteworthy impact. The topics include the analysis of the complexity of exponential time algorithms, constructing gadgets using computers and more. This article is more oriented to the problems of the complexity of Boolean functions. In addition, we give several open problems that we believe to be interesting, doable and fun.

The organization of this article is as follows: In Section 2, we start with the problem on the OBDDs as an illustrative example so that a large amount of computations lead us to a better understanding of the complexity. Then in Section 3, we consider three topics on Boolean circuits and see how actual computations can help to obtain theoretical results. In Section 4, we consider the problems on the expressive power of real valued polynomials for representing Boolean functions including a new computational method for getting an upper bound on the average density of polynomial threshold representations of Boolean functions.

2 Ordered Binary Decision Diagrams

We begin this article by reviewing the problem on the expressive power of ordered binary decision diagrams (OBDDs) and how computers are helping to give a better knowledge about the complexity. The ordered binary decision diagram is one of the most well studied models for representing Boolean functions both in theory and in practice.

Definition 1. *Let $X_n = \{x_1, \dots, x_n\}$ be a set of Boolean variables. A variable ordering π on X_n is a permutation from $\{1, \dots, n\}$ to X_n leading to the ordered list $\pi(1), \dots, \pi(n)$ of the variables.*

A π-OBDD on X_n is a directed acyclic graph whose sinks are labeled by a constant 0 or 1 and whose inner nodes are labeled by Boolean variables from X_n. Each inner node has two outgoing edges, one of them labeled by 0, the other by 1. The edges between inner nodes have to respect the variable ordering π, i.e., if an edge leads from an x_i-nodes to an x_j-node, then $\pi^{-1}(x_i) < \pi^{-1}(x_j)$. A $\pi - OBDD$ computes a Boolean function $f : \{0, 1\}^n \to \{0, 1\}$ in the following way: An assignment $(a_1, \dots, a_n) \in \{0, 1\}^n$ to X_n defines a uniquely determined path from the root to one of the sinks. The label of the reached sink gives $f(a)$. The size of a π-OBDD is defined as the number of its nodes. The OBDD complexity of f is the minimum size of a π-OBDDs that computes f. A π-OBDD for some unspecified variable order is simply called OBDD.

The size of an OBDD for a given Boolean function is strongly depending on the variable order. For example, consider the function $f(x_1, \dots, x_{2n}) = (x_1 \lor x_2) \land (x_3 \lor x_4) \land \cdots \land (x_{2n-1} \lor x_{2n})$. If we use the ordering $(x_1 x_3 \cdots x_{2n-1} x_2 x_4 \cdots x_{2n})$, then we only need $2n + 2$ nodes to represent f; however we need 2^{n+1} nodes if the ordering $(x_1 x_2 \cdots x_{2n})$ is used.

In the theory of OBDDs, one of the most investigated functions is the middle bit of integer multiplication. This is a $2n$-variable Boolean function that represents the n-th bit (the least significant bit is counted as the first) of the product of two n bit numbers $(x_{n-1} \cdots x_0)$ and $(y_{n-1} \cdots y_0)$ specified by inputs. This function is the first practical function for which an exponential lower bound has been proven for every variable order [12].

The investigation of the OBDD size of the middle bit of integer multiplication as well as other bits has a long history. An excellent survey devoted to this topic was presented by Bollig [11] in the BEATCS column. The newest volume of the famous book of Knuth [31] also discusses this topic extensively.

The current best lower bound is $2^{\lfloor n/2 \rfloor}/61 - 4$ by Woelfel [55] and the current best upper bound is $2.8 \cdot 2^{6n/5}$ by the author and Maruoka [3]. The upper bound is achieved by the pairwise ascending variable order $\pi = (x_0, y_0, \dots, x_{n-1}, y_{n-1})$.

n	4	5	6	7	8	9	10	11	12
Size of OBDD	31	63	136	315	756	1717	4026	9654	21931
Size of qOBDD	39	72	156	348	797	1808	4106	9796	22151
$2^{6n/5} \approx$	28	64	147	338	776	1783	4096	9410	21619

Figure 1: The minimum size of OBDD or qOBDD for the middle bit of multiplication. The data for qOBDD are from [3] and for OBDD are from [30, 47]

In fact, we found this by computer calculations. Below we describe a short story explaining this.

For an ease of exposition, we consider a variant of OBDDs called quasi-reduced OBDDs (qOBDDs); OBDDs where all variables have to be tested on every path from the source to the sinks. The size of π-qOBDD is at most $n + 1$ times larger than the size of π-OBDD for a same π, i.e., both are essentially the same (especially when we consider a function having an exponential complexity like integer multiplication). By the following nice fact, the size of π-OBDD for a given function f is fully characterized by the number of different subfunctions of f obtained by fixing appropriate variables according to π.

Fact 1. *Let f be a Boolean function over the variable set $X = \{x_1, \ldots, x_n\}$. For $I \subseteq X$, let sub(f, I) denote the number of different subfunctions of f obtained by fixing all variables in $X \backslash I$. Then, the number of $\pi(i)$-nodes in an optimal π-qOBDD for f is equal to sub(f, I) with $I = \{\pi(i + 1), \ldots, \pi(n)\}$.*

This immediately implies that the size of an optimal qOBDD for f is given by

$$\min_{\mathcal{I}=\{I_0,\ldots,I_n\}} \sum_{0\le i\le n} \mathrm{sub}(f, I_i), \tag{1}$$

where the minimum ranges over all sequences of sets $\phi = I_0 \subset I_1 \subset \cdots \subset I_n = X$ with $|I_i| = i$. By a standard dynamic programming, we can compute the optimal size of qOBDD for a given n-variable function as well as an optimal variable ordering in time $O(n^2 3^n)$ [17]. Note that we can similarly compute an optimal OBDD by replacing the term $\mathrm{sub}(f, I_i)$ in Eq. (1) by $\mathrm{sub}_x(f, I_i)$ denoting the number of different subfunctions of f obtained by fixing $\{\pi(i + 1), \ldots, \pi(n)\}$ and essentially depends on x. Current computers are fast enough to carry out these computations for up to $n \sim 20$.

The empirical results shown in Fig. 1 are bit surprising. The OBDD (or qOBDD) complexity of the middle bit of integer multiplication seems very well proportional to $2^{6n/5}$. For example, in the case of qOBDD, the optimal variable orderings for $n = 10, 11, 12$ are

$$(x_3x_4x_5x_6y_3y_4y_5y_6x_2y_2x_1y_1x_7y_7x_8y_8x_0y_9x_9y_0),$$
$$(x_3x_4x_5x_6x_7y_4y_5y_6y_7y_3x_2y_2x_1y_1x_8y_8x_9y_9x_0y_{10}x_{10}y_0),$$
$$(x_2x_3x_4x_5x_6x_7y_4y_5y_6y_7y_3y_2x_1y_1x_8y_8x_9y_9x_{10}y_{10}x_0y_{11}x_{11}y_0),$$

See [3, 30, 47] for more optimal orderings. These are enough to inspire a hypothesis that the pairwise ascending order $(x_0, y_0, \ldots, x_{n-1}, y_{n-1})$ or its slight modification $(x_1, y_1, \ldots, x_{n-2}, y_{n-2}, x_0, y_{n-1}, x_{n-1}, y_0)$ is a good ordering. Once we have this, showing the $O(2^{6n/5})$ upper bound is an easy task using Fact 1. Note that very recently this upper bound is shown to be asymptotically optimal if we fix the ordering to the pairwise ascending [47].

To see whether the true OBDD complexity of the middle bit of integer multiplication is $\Theta(2^{6n/5})$ seems to be an interesting open question, which is also appeared as an “exercise" in the Knuth’s book [31]. In addition, the following general question would also be interesting for understanding the nature of multiplication.

Problem 1. *For each k, determine the asymptotic OBDD complexity of the k-th bit of integer multiplication and find an optimal variable ordering for representing it.*

We believe that computer experiments would also help to attack this problem since we can obtain a *catalog* of optimal representations up to a relatively large number of inputs, say $n \sim 20$. Again, consult [11] for a recent progress in this topic.

3 Boolean Circuits

In spite of a huge amount of effort, we have very little knowledge about the circuit complexity. In 80’s, exponential lower bounds have been shown on the size of *monotone* circuits for the clique function as well as the size of *constant depth* circuits for the parity function. However, we should say that these two results are still the most important achievement in this area so far. To this date, the largest lower bounds on the circuit size for a function in NP is $5n$ [26]. We are eager to get a new idea for proving a stronger lower bound.

As we see in the last section, it would be possible to get a new insight by examining a catalog of optimal circuits for small functions generated by computers. This was at least partially succeeded for OBDDs. Several attempts have been made also for Boolean circuits using SAT solvers [54, 28].

On a current technology, the maximum size of circuits that can feasibly be enumerated by a computer is around 10. For example, in the recent volume of the famous Knuth's book [30, Chap. 7.1.2], he gave the complete classification of all Boolean functions on up to five variables in terms of their circuit complexity. The hardest function among all 5-variable Boolean functions over the basis B_2 (which contains all 2-input functions) needs 12 gates. Interestingly, such a function is essentially unique. However, it would not be feasible to enumerate all circuits with 20 gates, even in a near future.

In this section, we review three another approaches aiming to use computers in proving lower bounds. The first two are to reduce the lower bound problem to a polynomially solvable optimization problem, and the last one is a graph theoretic approach based on the computer search. So far, these approaches could not deliver a significant lower bound. However, we hope that pushing them further would yield a new insight on how to prove a stronger lower bound on a stronger computational model.

3.1 Lower Bounds for Depth Two Threshold Circuits via LP

A major open problem in circuit complexity is to give a superpolynomial lower bound on the size of depth-2 threshold circuits (with unrestricted weights) for an explicit Boolean function. Many exponential lower bounds are known for depth-2 threshold circuits with various restrictions (see e.g., [49] and the references therein). Among these restricted circuits, we consider in this section depth-two circuits with a threshold gate at the top and symmetric gates below. Such circuits have been considered before in e.g., [15]. Below we demonstrate that an exponential lower bound for this model can be obtained by solving a large-scale linear program using LP solvers. Note that the contents of this section is an updated version of [2] that was built on the work by Basu et al. [9].

Let $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$ be two binary inputs of length n. The *inner product mod 2* function, denoted by $\mathsf{IP}_n(X, Y)$, is defined as $\oplus_i x_i y_i$ where $\oplus$ denotes the exclusive-OR operation.

A linear threshold function $f(X)$ is a Boolean function on input $X = (x_1, \ldots, x_n) \in \{0, 1\}^n$ such that

$$f(X) \quad = \quad sgn\left(w_0 + \sum_{i=1}^{n} w_i x_i\right),$$

where $w = (w_0, \ldots, w_n) \in \mathbb{R}^{n+1}$ is called the *weights*, and *sgn* stands for the sign function: $sgn(x) = 1$ if $x > 0$, and $sgn(x) = 0$ otherwise. A *threshold gate* is a gate that computes a threshold function. A function $f : \{0, 1\}^X \to \mathbb{R}$ is called *symmetric* if the value of f depends only on the number of inputs that are 1. A *symmetric gate* is a gate that computes a symmetric function. For a Boolean function f, the minimum number of symmetric gates in a depth-two circuit of "threshold-of-symmetric gates" that computes f is denoted by $s(f)$.

It is convenient to consider a polynomial P of the form

$$P(X,Y) \;=\; \sum_{S \subseteq X \cup Y} w_S h_S(X,Y), \tag{2}$$

where $w_S \in \mathbb{R}$ and h_S denote a symmetric function over the variable set S. The *support* of P is defined as $\{S \subseteq X \cup Y \mid w_S \neq 0\}$. We say that P *sign-represents* f if $P(X) > 0$ whenever $f(x) = 1$ and $P(X) < 0$ whenever $f(x) = 0$. Obviously, $s(f)$ is equal to the minimum size of the support of a polynomial that sign-represents f.

Let f be a (not necessarily Boolean) function on a set of variables X and ρ be a partial assignment of the variables, i.e., ρ is a map from X to the set $\{0, 1, *\}$. The restriction of f by ρ, denoted by $f|_\rho$, is the function obtained from f by setting x_i to be $\rho(x_i)$ if $x_i \in \{0, 1\}$ and leaving x_i if $x_i = *$. For a partial assignment ρ, let $res(\rho)$ denote the set of variables that mapped to 0 or 1 by ρ.

We also define the restriction of a polynomial P of the form (2) by ρ, denoted by $P|_\rho$ as follows: First, replace each h_S in P by $h_S|_\rho$. Note that $h_S|_\rho$ is a symmetric function on $S \backslash res(\rho)$. Then, for every S_1 and S_2 such that $h_{S_1}|_\rho$ and $h_{S_2}|_\rho$ are on the same set of variables S', then replace $w_{S_1}h_{S_1}|_\rho + w_{S_2}h_{S_2}|_\rho$ by an equivalent symmetric function $h'_{S'}$. This is always possible since the sum of two symmetric functions is also a symmetric function.

Suppose that P is an optimal polynomial that sign-represents IP_n. Consider two assignments $\alpha : (x_1, y_1) = (0, 1)$ and $\beta : (x_1, y_1) = (1, 1)$. Since $P|_\alpha$ sign-represents IP_{n-1} and $P|_\beta$ sign-represents the *complement* of IP_{n-1}, it is obvious that the polynomial $P|_\alpha - P|_\beta$ sign-represents IP_{n-1}.

We now divide P into two subformulas P_0 and P_1; P_1 is consisting of all terms including x_1, and P_0 is the rest. Let $\sharp(P)$ denote the number of terms in P. We have

$$P|_\alpha - P|_\beta \;=\; P_0|_\alpha + P_1|_\alpha - P_0|_\beta - P_1|_\beta = P_1|_\alpha - P_1|_\beta,$$

since P_0 is independent of x_1. This implies $\sharp(P_1) \geq \sharp(P_1|_\alpha - P_1|_\beta) \geq s(\mathsf{IP}_{n-1})$. The first inequality follows from a simple observation. Since $s(\mathsf{IP}_n) = \sharp(P_0) + \sharp(P_1)$, if we could similarly show that $\sharp(P_0) \geq s(\mathsf{IP}_{n-1})$, then we would get the recursion $s(\mathsf{IP}_n) \geq 2s(\mathsf{IP}_{n-1})$ which immediately gives a lower bound of $s(\mathsf{IP}_n) \geq 2^n$. However, we cannot cancel out $P_1|_{\sigma_1} - P_1|_{\sigma_2}$ by any two assignments σ_1 and σ_2 to $\{x_1, y_1\}$.

Instead, we consider assignments to *four* variables $\{x_1, x_2, y_1, y_2\}$ and divide P into $2^4 = 16$ parts depending on the intersection of these four variables and the support of monomials. For $T \subseteq \{x_1, y_1, x_2, y_2\}$, let P_T be a subformula of P consisting of all terms $w_S h_S$ such that $S \cap \{x_1, y_1, x_2, y_2\} = T$.

Consider two assignments $\alpha : (x_1, y_1, x_2, y_2) = (0,1,1,0)$ and $\beta : (x_1, y_1, x_2, y_2) = (1,1,0,0)$. Obviously, $P|_\alpha - P|_\beta$ sign-represents IP_{n-2}. Here we have

$$P|_\alpha - P|_\beta = \sum_{T \subseteq \{x_1,y_1,x_2,y_2\}} (P_T|_\alpha - P_T|_\beta) = \sum_{T:|T\cap\{x_1,x_2\}|=1} (P_T|_\alpha - P_T|_\beta),$$

since polynomials $P_T|_\alpha - P_T|_\beta$ are canceling out when $|T \cap \{x_1, x_2\}| = 0$ or 2.

We introduce new variables q_T's that represent $\sharp(P_T)/s(\mathsf{IP}_{n-2})$. Then the above equation implies the linear inequality

$$\sum_{T:|T\cap\{x_1,x_2\}|=1} q_T \geq 1. \tag{3}$$

Eight variables (out of 16) are appeared in the LHS of the above inequality. On the other hand, since $\sharp(P) = \sum_T \sharp(P_T)$ we have

$$\sum_{T\subseteq\{x_1,y_1,x_2,y_2\}} q_T = \frac{s(\mathsf{IP}_n)}{s(\mathsf{IP}_{n-2})}.$$

If we consider another pair of assignments, then we get another inequality similar to Ineq. (3). By considering four pairs of Type 1 assignments, and four pairs of Type 2 assignments described below, we get the *system* of inequalities shown in Fact 2.

Type 1 Choose $i \in \{1,2\}$ and $v \in \{x_i, y_i\}$. The unchosen variable in $\{x_i, y_i\}$ is denoted by u. Let $\alpha : (v,u) = (0,1)$ and $\beta : (v,u) = (1,1)$.

Type 2 Choose $v_1 \in \{x_1, y_1\}$ and $v_2 \in \{x_2, y_2\}$. Let u_1 and u_2 be the unchosen variables in $\{x_1, y_1\}$ and in $\{x_2, y_2\}$, respectively. Let $\alpha : (v_1, u_1, v_2, u_2) = (0,1,1,0)$ and $\beta : (v_1, u_1, v_2, u_2) = (1,1,0,0)$.

Fact 2. *Let z be the minimum value of the objective function of the following linear program. Then $s(\mathsf{IP}_n) \geq z \cdot s(\mathsf{IP}_{n-2})$.*

$$\begin{array}{lll} \textit{Minimize} & \displaystyle\sum_{T\subseteq\{x_1,y_1,x_2,y_2\}} q_T & \\ \textit{Subject to} & \displaystyle\sum_{T:v\in T} q_T \geq 1 & (v \in \{x_1,y_1,x_2,y_2\}) \\ & \displaystyle\sum_{T:|\{v_1,v_2\}\cap T|=1} q_T \geq 1 & (v_1 \in \{x_1,y_1\}, v_2 \in \{x_2,y_2\}), \\ & q_T \geq 0 & (T \subseteq \{x_1,x_2,y_1,y_2\}). \end{array} \tag{4}$$

LP (4) has 2^4 variables and 8 constrains, and is easy to solve. The minimum value of the objective function is 1.5 which implies $s(\mathsf{IP}_n) \geq 1.5^{n/2} \sim 1.2247^n$.

Quite naturally, a lower bound is improved by considering more assignments. Let $k \geq 3$ be an integer. We consider a set of pairs of assignments on $\{x_1, y_1, \ldots, x_k, y_k\}$ of the following two types.

Type 1 Choose $i \in \{1, \ldots, k\}$ and $v \in \{x_i, y_i\}$. The unchosen variable in $\{x_i, y_i\}$ is denoted by u. Let $\alpha : (v, u) = (0, 1)$ and $\beta : (v, u) = (1, 1)$.

Type 2 Choose $i, j \in \{1, \ldots, k\}$ with $i \neq j$. Choose $v_1 \in \{x_i, y_i\}$ and $v_2 \in \{x_j, y_j\}$. Let u_1 and u_2 be the unchosen variables in $\{x_i, y_i\}$ and in $\{x_j, y_j\}$, respectively. Let $\alpha : (v_1, u_1, v_2, u_2) = (0, 1, 1, 0)$ and $\beta : (v_1, u_1, v_2, u_2) = (1, 1, 0, 0)$.

Note that for two assignments α and β of Type i ($i \in \{1, 2\}$), $P|_\alpha - P|_\beta$ sign-represents IP_{n-i}. By dividing P into 2^{2k} parts and letting q_T be $\sharp(P_T)/s(\mathsf{IP}_{n-k})$, we can show that:

Fact 3. *Suppose that $k \geq 3$. Let z_{k-1} and z_{k-2} be real numbers such that $s(\mathsf{IP}_n) \geq z_{k-1} \cdot s(\mathsf{IP}_{n-(k-1)})$ and $s(\mathsf{IP}_n) \geq z_{k-2} \cdot s(\mathsf{IP}_{n-(k-2)})$ for every n. Let z_k be the minimum value of the objective function of the following linear program. Then $s(\mathsf{IP}_n) \geq z_k \cdot s(\mathsf{IP}_{n-k})$.*

$$\begin{array}{llll} \textit{Minimize} & \displaystyle\sum_{T \subseteq \{x_1, y_1, \ldots, x_k, y_k\}} q_T & & \\ \textit{Subject to} & \displaystyle\sum_{T : v \in T} q_T & \geq z_{k-1} & (v \in \{x_1, y_1, \ldots, x_k, y_k\}) \\ & \displaystyle\sum_{T : |\{v_1, v_2\} \cap T| = 1} q_T & \geq z_{k-2} & \left(\begin{array}{l} i, j \in \{1, \ldots, k\}, i \neq j \\ v_1 \in \{x_i, y_i\}, v_2 \in \{x_j, y_j\} \end{array} \right) \\ & q_T & \geq 0 & (T \subseteq \{x_1, y_1, \ldots, x_k, y_k\}). \end{array} \tag{5}$$

Note that the constraint matrix of LP (5) is a $(2k + 4\binom{k}{2}) \times 2^{2k}$ binary matrix and easy to generate by a simple computer program. In addition, if the value of k is relatively small, then we can solve this by an LP solver.

Solving LP (5) for $k = 3$ with $z_1 = 1$ and $z_2 = 1.5$ yields $z_3 = 2$. This implies $s(\mathsf{IP}_n) \geq 2^{n/3} \sim 1.2599^n$, which is slightly better than the lower bound obtained by solving LP (4). Solving LP (5) again for $k = 4$ with $z_2 = 1.5$ and $z_3 = 2$ yields $z_4 \sim 2.8333$, which implies better lower bound of $s(\mathsf{IP}_n) \geq 2.8333^{n/4} \sim 1.2974^n$. By repeating this procedure, we can obtain $z_5 \sim 4.0277$, $z_6 \sim 5.7500$, $z_7 \sim 8.2541$, $z_8 \sim 11.9700$ and $z_9 \sim 17.3350$. These imply the lower bounds on $s(\mathsf{IP}_n)$ of 1.3213^n, 1.3384^n, 1.3519^n, 1.3638^n and 1.3729^n, respectively. We have not succeeded to compute the value of z_k for $k \geq 10$ at the time of writing this article (because GLPK solver [35] is killed by the out of memory).

Note that the best known lower bound on $s(\mathsf{IP}_n)$ is $\Omega(2^{n/2}/n) = \Omega(1.4142^n)$ by Forster et al. [15] and the upper bound is $s(\mathsf{IP}_n) \leq 2^n$. The lower bound is proved by considering the rank of a communication matrix that sign-represents IP_n [15]

(see also [49] for a generalization). The upper bound follows from the construction $\mathsf{IP}_n(X, Y) = \mathrm{sign}(\sum_{S\subseteq[n]}(-2)^{|S|+1}X_S Y_S)$, where X_S and Y_S denote $\prod_{i\in S} x_i$ and $\prod_{i\in S} y_i$, respectively.

At this moment, we don't know whether our method can beat $\Omega(1.4142^n)$. However, we think that there is a chance. By examining LP (5) with $k = 8$ under the assumption $s(\mathsf{IP}_n) \geq \sqrt{2} \cdot s(\mathsf{IP}_{n-1})$ for every n, we obtain somewhat curious fact saying that the lower bound would be *enhanced* by considering a large LP.

Fact 4. *Suppose that, for every sufficiently large n,* $s(\mathsf{IP}_n) \geq \sqrt{2} \cdot s(\mathsf{IP}_{n-1})$ *holds. Then* $s(\mathsf{IP}_n) = \Omega(1.4198^n)$.

3.2 Lower Bounds on Formula Size via SDP

A *Boolean formula* is a binary tree where each internal node is labeled with $\wedge$ or $\vee$, and each leaf is labeled with a literal, i.e., a variable or its negation. A Boolean formula computes a Boolean function in an obvious way. The *size* of a formula is the number of leaves in the tree. For a Boolean function f, the *formula complexity*, denoted by $L(f)$, is defined as the size of a smallest formula that computes f. The famous result of Khrapchenko [27] says that the formula complexity of the parity of n variables is at least n^2. The current best lower bound for an explicitly defined function is $n^{3-o(1)}$ due to Håstad [21].

The quantum adversary method ([5],[8],[33],[56]) has originally been developed for proving lower bounds on quantum query complexity. Laplante, Lee and Szegedy [32] revealed that this method is also very useful for lower bounding the formula size. In this framework, a lower bound on the formula size of an n-variable Boolean function can be obtained by solving an SDP (semidefinite program) of the order $n2^n$.

Let Γ be a $2^n \times 2^n$ be a Hermitian matrix with rows and columns labeled by elements of $\{0, 1\}^n$ such that $\Gamma[x, y] = 0$ whenever $f(x) = f(y)$. Let $\|M\|$ denote the spectral norm of the matrix M. For a Boolean function $f : \{0, 1\}^n \to \{0, 1\}$, the adversary bound for f is defined as

$$\mathrm{ADV}(f) \quad = \quad \max_{\Gamma\geq 0,\Gamma\neq 0} \frac{\|\Gamma\|}{\max_i \|\Gamma \circ D_i\|},$$

where the maximum is taken over nonnegative symmetric matrices Γ, and D_i is a zero-one matrix where $D_i[x, y] = 1$ iff $x_i \neq y_i$. $\Gamma \circ D_i$ denotes the entry-wise product of Γ and D_i.

Laplante, Lee and Szegedy [32] proved that $\mathrm{ADV}(f)^2$ is a lower bound on the formula complexity of f. This parameter can be formulated as SDP [52]: Let F be a $2^n \times 2^n$ binary matrix such that $F[x, y] = 1$ iff $f(x) \neq f(y)$, and let D_i be defined as above. The parameter ADV(f) is given by $1/\mu_{min}$, where μ_{min} is the minimal solution of the following semidefinite program:

$$\begin{array}{lrcl} \text{Minimize} & \mu = \mathrm{tr}\Delta & & \\ \text{Subject to} & \Delta & \text{is} & \text{diagonal}, \\ & Z & \geq & 0, \\ & Z \cdot F & = & 1, \\ & \forall i : \Delta - Z \circ D_i & \geq & 0, \end{array} \tag{6}$$

Here trΔ denotes the trace of a matrix Δ.

This parameter enjoys a nice composition property. For two Boolean functions f on n variables and g on m variables, let $f \otimes g$ denote a composite function on nm variables: $(f \otimes g)(x_1, \ldots, x_{mn}) = f(g(\tilde{x_1}), \ldots, g(\tilde{x_n}))$ where $\tilde{x_i} = (x_{(i-1)m+1}, \ldots, x_{im})$.

Theorem 1. *([5, 22]) For every Boolean functions f and g,* ADV$(f \otimes g)$ = ADV$(f) \cdot$ ADV(g).

This theorem says that if $L(f) = \mathrm{ADV}(f)^2$ and $L(g) = \mathrm{ADV}(g)^2$, then $L(f \otimes g) = L(f) \cdot L(g)$. This gives a nice generalization of the result of Khrapchenko [27]. The parity function on $n = 2^k$ variables can be written as $(x_1 \oplus x_2) \otimes \cdots \otimes (x_1 \oplus x_2)$. The optimal formula for $x_1 \oplus x_2$ is $(x_1 \wedge \overline{x_2}) \vee (\overline{x_1} \wedge x_2)$, which has size 4. By using this recursively, we get a formula for the parity on 2^k variables whose size is $4^k = n^2$. Khrapchenko's n^2 lower bound for the parity function guarantees the exact optimality of such a naive construction. Note that, recently, Tarui [53] proved that the formulas constructed in this way are essentially unique smallest ones.

Theorem 1 shows a similar optimality can be established for any base functions f with the property $L(f) = \mathrm{ADV}(f)^2$. We call such a function *tight*. By using an SDP solver, we can enumerate all tight functions on up to five variables. We say that two Boolean functions are in the same NPN-equivalence class if they can be equivalent by negating of output and input variables and permuting of input variables. The number of NPN-equivalence classes of $2, 3, 4$ and 5 or fewer variables are $2, 14, 222$ and 616126, respectively. Out of them, $4, 8, 20$ and 55 classes are tight [23, 18]. Example of such functions are $\mathsf{MUX}(x_1, x_2, x_3)$ and $\mathsf{MUX}(x_1 \oplus x_2, x_3 \oplus x_4, x_3 \oplus x_5)$, where $\mathsf{MUX}(x_1, x_2, x_3)$ denotes the multiplexer function $x_1x_2 \vee \overline{x_1}x_3$. At this moment, we don't know whether the tightness is a *necessary* condition for such an optimality result.

It should be noted that unfortunately, it is known that $\mathrm{ADV}(f)^2$ is upper bounded by n^2. See also [24] for another explanation of the limit of this method. So we should in mind that this method can not yield a super-quadratic lower bound on the formula size.

Note also that Høyer, Lee and Sp̌alek [23] strengthen this method. They introduced new parameter

$$\mathrm{ADV}^{\pm}(f) \quad = \quad \max_{\Gamma \neq 0} \frac{\|\Gamma\|}{\max_i \|\Gamma \circ D_i\|},$$

in which the condition $\Gamma \geq 0$ in ADV(f) is removed, and proved that $\mathrm{ADV}^{\pm}(f)^2$ is also a lower bound on formula size. The computational results for this parameter as well as the program for MatLab can be found on the web page in the reference of [23].

In the preceding two sections, we reviewed lower bound problems on two computational models can be formulated as linear program and semidefinite program. Thus, an interesting question is:

Problem 2. *Can we formulate a lower bound problem on a stronger model as a polynomial time solvable optimization problem?*

3.3 Linear Lower Bounds on Circuit Size

Let B_2 denote the set of all (sixteen) Boolean functions over two variables, and let U_2 denote $B_2 \backslash \{\oplus, \equiv\}$, i.e., B_2 excluding the parity and its negation. For a Boolean function f, the B_2-circuit complexity (U_2-circuit complexity, resp) of f is the minimum size of a circuit over B_2 (U_2, resp.) that computes f. After a long line of research for improving a constant factor of linear lower bounds on circuit complexity, the current best lower bound on B_2-circuit complexity for a function in NP is $3n$ [10] and that on U_2-circuit is $5n$ [26] (see also [34, 25]). Essentially, all linear lower bounds are proved by so called the *gate-elimination method*.

A typical proof using the gate-elimination method is as follows: First we define some property of n-variable Boolean functions $Q(n)$. Then we show that, for an optimal circuit for a function having $Q(n)$, a specific number of gates, say α, can be eliminated by fixing some variable to a constant 0 or 1 and the resulting function has the property $Q(n-1)$. Applying this inductively gives an αn lower bound on a circuit size for a function having $Q(n)$.

The core of the proof is typically by the case analysis on the “local pattern" of circuits near the input terminals. That became more and more complicated as the constant factor increases; the proof of $5n$ lower bounds by Iwama et al. [26] needs to analyze several dozen of cases in which the deepest one is like Case 2.3.3.3.4.

Such an analysis would be automated by computers. To examine the possibility of this approach, we briefly review their lower bound proof.

Let C be a circuit over $X_n = \{x_1, \ldots, x_n\}$. Recall that a partial assignment σ is a mapping from X_n to $\{0, 1, *\}$. We say that the set of variables fixed to a constant by σ the *support* of σ. For a Boolean function f on X_n and a partial assignment

σ, let $f|_\sigma$ denote the function obtained from f by applying σ. We use the similar notation $C|_\sigma$ for a Boolean circuit C.

The proof by Iwama et al. [26] (as well as $4.5n$ bound by Lachish and Raz [34]) uses the following property of Boolean functions:

Definition 2. *A Boolean function f over $X_n = \{x_1, \ldots, x_n\}$ is k-*mixed *if for every $V \subseteq X_n$ such that $|V| = k$ and for any two distinct partial assignments α and β with support V, the functions obtained from f by applying α and β are distinct, i.e., $f|_\alpha \neq f|_\beta$.*

Note that they originally used the property called *strongly-two-dependent*; but the property of k-mixed is stronger than this.

The main portion of their proof is the following lemma.

Lemma 1. *Let f be a t-mixed n-variable Boolean function with $n-t = k$. Suppose that $n \geq 2k+4$ and a circuit C computes f. Then, there exists a partial assignment σ that satisfies the following: $|\sigma| \leq 2$, and there exist a Boolean circuit $C' \equiv C|_\sigma$ such that $\mathrm{SD}(C) \geq \mathrm{SD}(C') + 5|\sigma|$. Here $\mathrm{SD}(C)$ is a measure of a circuit size which is a slight modification of the normal "number of gates" measure (see [26] for the definition).*

By using above lemma inductively we can obtain the lower bound of $5n - o(n)$ for every k-mixed function with $k = n - o(n)$.

Such a method can be reformulated as follows: In order to show the αn lower bound on a circuit complexity of f, it is sufficient to give a class of *local circuit patterns $\mathcal{D}$* that satisfies the following.

(C1) For every optimal circuit C that computes f, some pattern $D \in \mathcal{D}$ appears in C.

(C2) For every circuit C and every $D \in \mathcal{D}$, if D appears in C, then there exists a partial assignment σ and a circuit $C' \equiv C|_\sigma$ such that

$$[\text{the size of } C] - [\text{the size of } C'] \geq \alpha|\sigma|.$$

Interestingly, this formulation of the gate-elimination method is quite resemble to the famous proof of the Four Color Theorem [6, 7, 45]. The computer assisted proof of this theorem has given by Appel and Haken three decades ago. A simplified but still computer assisted proof has then given by Robertson et al. [45] which proceeds as follows.

Suppose for the contrary that there exists a counterexample to the Four Color Theorem. Let P be some easily describable graph theoretic property that a minimal counterexample to the Four Color Theorem has. Then, they exhibited a set $\mathcal{D}$ of 633 "configurations", that consists of small graphs with some additional information, satisfying the following:

(F1) For every planar graph G with property P, some configuration $D \in \mathcal{D}$ appears in G.

(F2) If a configuration $D \in \mathcal{D}$ appears in G, then G is not a minimal counterexample to Four Color Theorem.

By verifying above two statements using computers, we can conclude that no minimal counterexample exists, and hence Four Color Theorem is true.

We now review the propositions ($C1$) and ($C2$). Suppose that a circuit C is a minimal counterexample to a lower bound, i.e., C is an optimal circuit that computes f and the size of it is less than αn. Recall that a circuit is defined as a graph in which nodes represent gates and edges represent wires. We can put some graph theoretic properties for a circuit C by its optimality, which we consider as the property P in (F1). Then (C2) says that if C contains $D \in \mathcal{D}$ then, C is not a minimal counterexample (since if C is a counterexample, then a smaller circuit $C|_\sigma$ is also).

It would be interesting to give a (minimal) set of circuit patterns that yield a lower bound of $5n$ or even higher. We have in fact tried to pursuit this approach for getting a higher lower bound, but failed. However, during the experiments, the author and Tarui [4] have succeeded(?) to find the reason why we have failed.

In fact, there *exists* a k-mixed Boolean function with $k = n - o(n)$ that can be computed by a U_2-circuit of size $5n + o(n)$. The $5n$ lower bounds for this class of functions have already been tight.

Below we sketch the construction of the function, which is a modification of a function introduced by Savický and Žák [48].

Let p be a prime such that $n \leq p < 2n$. Define $w_n : \{0, 1, \ldots\} \to \{1, \ldots, n\}$ so that $w_n(s)$ is the residue of s modulo p, if this residue lies in $\{1, \ldots, n\}$, and is 1 otherwise. Put $b = \lceil \frac{n}{\lceil \log n^2 \rceil} \rceil$. We split the interval $\{1, \ldots, n\}$ into b blocks $D_1, \ldots, D_b$ of equal size. For every $n \in \mathbb{N}$, $f_n(x_1, \ldots, x_n)$ outputs one of its input x_z where an index is given by

$$z = w_n\left(\sum_{i=1}^{b}\left(i \cdot \bigoplus_{j \in D_i} x_j\right)\right).$$

Theorem 2. *[4] The function f_n defined above is k-mixed with $k = n - \omega(\sqrt{n}\log^2 n)$ and can be computed by a U_2 circuit of size $5n + o(n)$.*

This result means that every lower bounds method that applies to arbitrary k-mixed functions cannot show a lower bound higher than $5n$. So the problem we should tackle first is:

Problem 3. *Find a property of Boolean functions that can be used to derive a lower bound higher than 5n.*

We still believe that a graph theoretic approach described above helps to inspire such a property.

4 Polynomial Representation of Boolean Functions

In this section, we consider two problems both are on the expressive power of real polynomials for representing Boolean functions. This is one of the most active subjects in the research of computational complexity. See e.g., [50, 51] for a recent progress on this topic.

4.1 Degree of Full-Sensitive Functions

There have been developed many complexity measures for Boolean functions, and investigated the relation among them. See e.g., a good survey by Buhrman and de Wolf [13]. In this section, we consider the relationship between two major measures; the sensitivity and the degree of Boolean functions.

Let f be an n-variable Boolean function. The *sensitivity* of f on x is the number of bit positions i such that $f(x) \neq f(x^i)$, where x^i denotes x with its i-th bit flipped. The *sensitivity* of f is $s(f) = \max_x s_x(f)$.

An n-variable polynomial $p : \mathbb{R}^n \to \mathbb{R}$ *represents* f is $p(x) = f(x)$ for every $x \in \{0, 1\}^n$. Note that each Boolean function can be represented by a *unique* multivariate polynomial (see e.g., Lemma 1 in [13]). The degree of f is the degree of this multivariate polynomial that represents f.

Definition 3. *An n-variable Boolean function f is* fully sensitive *if the sensitivity of f is n.*

We consider the following simple question.

Problem 4. *What is the minimum degree of an n-variable Boolean function that is fully sensitive?*

The current best lower bound is $\sqrt{n/2}$ and the upper bound is $O(n^{\log_6 3}) = O(n^{0.613})$. One of the importance of the problem of finding a fully sensitive and low degree function is that it gives the largest known gap between the decision tree complexity and the "log-rank" of the communication matrix (see e.g., [29] and [39]).

The lower bound is proved by Nisan and Szegedy [38]. Note that if we allow some approximation, this lower bound is asymptotically tight. There is a polynomial p of degree $O(\sqrt{n})$ such that $|p(x_1, \ldots, x_n) - OR(x_1, \ldots, x_n)| \leq 1/3$ for every x, where OR denotes the OR function that is apparently fully sensitive at the origin $x = 00 \cdots 0$ (see [38, Example 3.11] for the construction using Chebyshev polynomials). Note also that the degree of the OR function is exactly n.

Nisan and Szegedy [38] have also gave the construction of a fully sensitive function whose degree is $n^{\log_3 2} \sim n^{0.631}$. Let E_2 denote the Boolean function on three variables given by

$$E_2(x_1, x_2, x_3) \quad = \quad x_1 + x_2 + x_3 - x_1x_2 - x_1x_3 - x_2x_3.$$

It outputs 1 iff one or two of its inputs are 1. Define $E_2^{(k)}$ as the function on $n = 3^k$ variables obtained by building a complete ternary tree of depth k, where the 3^k leaves are the variables and each node is the E_2-function of its three children. Then $E_2^{(k)}$ is represented by $E_2(E_2^{(k-1)}, E_2^{(k-1)}, E_2^{(k-1)})$, and so by a polynomial of degree $2^k = n^{\log_3 2}$. It is easy to check that $E^{(k)}$ is fully sensitive at the origin.

It is noticeable that, in the above construction, we can use any "base" function if it is fully sensitive at the origin. If we use an n-variable function of degree k, then we can get a fully sensitive N-variable function of degree $N^{\log_n k}$.

Along to this line, Kushilevits has improved the exponent to $\log_6 3 = 0.613...$, which is the current best, by exhibiting such a function on 6 variables of degree 3 (see [39, footnote 1 on p.560]).

Let $S(6)$ be a family of 3-sets on $\{1, 2, \dots, 6\}$ defined as

$$\begin{aligned} S(6) \quad = \quad & \{(1,2,3),(1,2,4),(3,4,5),(3,4,6),(1,5,6), \\ & \quad (2,5,6),(1,3,5),(1,4,6),(2,3,6),(2,4,5)\}. \end{aligned}$$

Let F_3 be a polynomial on 6 variables defined as

$$F_3(x_1, \dots, x_6) \quad = \quad \sum_{i:1 \le i \le 6} x_i - \sum_{(i,j):1 \le i < j \le 6} x_ix_j + \sum_{(i,j,k) \in S(6)} x_ix_jx_k.$$

Interestingly, $S(6)$ is also appeared in a major open question in *extremal graph theory*.

Given an r-graph $\mathcal{F}$ on a vertex set V, i.e., a hypergraph whose edge set consists of r-sets of V, the *Turán number* $\mathrm{ex}(n, \mathcal{F})$ is the maximum number of edges in an n-vertex r-graph not containing a copy of $\mathcal{F}$. The *Turán density* of $\mathcal{F}$ is defined as

$$\pi(\mathcal{F}) \quad = \quad \lim_{n \to \infty} \frac{\mathrm{ex}(n, \mathcal{F})}{\binom{n}{r}}.$$

Determining $\pi(\mathcal{F})$ for a given $\mathcal{F}$ is a notoriously hard problem (see e.g., [36, 44] and the references therein). One of the most well-investigated cases is $\pi(\mathcal{K}_4^-)$, where $\mathcal{K}_4^- = \{abc, abd, acd\}$ is the complete 3-graph on 4 vertices with one edge removed. It is known that $\pi(\mathcal{K}_4^-) \geq 2/7$, and this lower bound is strongly believed to be tight.

Obtaining an upper bound on the density is very difficult. Recently, Razborov [43, 44] developed an intriguing technique called "Flag Algebra" that can be used to attack this problem. Very roughly speaking, in this framework, the problem of upper bounding the Turán density is reduced to the feasibility of some well-designed semidefinite programming problems. Designing a good problem is typically relying on a computer calculation. This seems to be another domain that computers can help in obtaining theoretical results. In [44], it is described that the current best upper bound $\pi(\mathcal{K}_4^-) \leq 0.2978$ can be obtained by this method.

The lower bound $\pi(\mathcal{K}_4^-) \geq 2/7$ follows from the construction due to Frankl and Füredi [16] that "blow-ups" the system $S(6)$ (see [16] for the detail). This construction gives a sequence of $\mathcal{K}_4^-$-free 3-graphs with asymptotic density 2/7. In fact, the problem to find an extremal $\mathcal{K}_4^-$-free graph is formulated as a 0-1 integer programming problem on the variable set $\{x_S \mid S \subseteq V \text{ and } |S| = 3\}$:

$$\begin{array}{lll} \text{Maximize} & \sum_S x_S & \\ \text{Subject to} & \sum_{S \subseteq T} x_S \leq 2 & (\forall T \subseteq V \text{ with } |T| = 4), \end{array}$$

It is easy to check that $S(6)$ is the unique solution of this problem for $n = 6$. Markström and Talbot [36] conducted a systematic search using a computer and provided all extremal $\mathcal{K}_4^-$-free 3-graphs on n vertices for $n \leq 19$. Quite interestingly, for every $11 \leq n \leq 19$, the unique extremal graph is this blow-up construction. Motivating by this, they conjectured that this is true for every $n \geq 11$, i.e., the recursive use of a small gadget would always give an optimal construction.

Similar to this problem, our problem can also be written as an integer programming problem. Note that every coefficient of a polynomial that represents a Boolean function must be an integer. The variable set is $\{x_S \mid S \subseteq \{1, \ldots, n\}$ and $|S| \leq k\}$, and the existence of a fully sensitive degree k function on n variables is given by the feasibility of the following system (without objective function):

$$\begin{array}{lll} \text{Subject to} & 0 \leq \sum_{S \subseteq T} x_S \leq 1, & (\forall T \subseteq V), \\ & x_S = 1, & (\forall S \text{ with } |S| = 1), \\ & x_\phi = 0. & \end{array} \tag{7}$$

It is natural to expect that a good formula can be obtained by solving this problem by IP solvers. A simple examination can verify that F_3 is the unique polynomial (up to a permutation and negation of variables) of degree 3 that represents a fully sensitive Boolean function on 6 variables. In addition, there are no such polynomials of degree 3 on 7 variables.

We start with the case of degree 4. We already have such a construction for $n = 9$, which is $E_2^{(2)}$. Again, this construction is optimal. We below give a (fully theoretical) proof verifying this since it says a bit more than what the well-known symmetrization technique (Fact 5) introduced by Minsky and Papert [37] can say. In the sequel, we denote the number of ones in a binary vector x by $|x|$.

Fact 5. *Let f be a Boolean function on n variables, and p be a polynomial that represents f. Then there exists a univariate polynomial $\tilde{p}$ such that (i) for every $k \in \{0, 1, \ldots, n\}$,*

$$\tilde{p}(k) \quad = \quad a_0 + a_1\binom{k}{1} + a_2\binom{k}{2} + \cdots + a_n\binom{k}{n}, \tag{8}$$

where $a_i = \alpha_i/\binom{n}{i}$ and α_i is the sum of the coefficients of all degree i monomials in p, (ii) $\tilde{p}(k)$ is equal to the probability that $f(x)$ outputs 1 when x is chosen uniformly from all input vectors with Hamming weight k.

Proof. Consider a real valued polynomial

$$p^{sym}(x_1, \ldots, x_n) \quad = \quad \frac{\sum_\pi p(x_{\pi(1)}, \ldots, x_{\pi(n)})}{n!},$$

where the summation is over all permutations π on $\{1, \ldots, n\}$. Since p^{sum} only depends on the number of ones in inputs, there exist a unique univariate polynomial $\tilde{p}$ such that

$$\tilde{p}(x_1 + \cdots + x_n) \quad = \quad p^{sym}(x_1, \ldots, x_n).$$

It is easy to check that $\tilde{p}$ satisfies the conditions in the fact. □

Theorem 3. *There are no Boolean functions f on* 10 *variables such that f is fully sensitive and the degree of f is at most* 4.

Proof. Suppose for the contrary that a multilinear polynomial p of degree 4 on $X = \{x_1, \ldots, x_{10}\}$ represents a f such that f is fully sensitive. W.l.o.g., this happens at the origin. This immediately implies that (i) every monomial of degree 1 in p has the coefficient 1, and (ii) every monomial of degree 2 has the coefficient -1 or -2. By applying the symmetrization (Eq. (8)), we have

$$\tilde{p}(k) \quad = \quad k + a_2\binom{k}{2} + a_3\binom{k}{3} + a_4\binom{k}{4}.$$

for some a_2, a_3 and a_4. By (ii) of Fact 5, $0 \le \tilde{p}(k) \le 1$ for every $k = 0, \ldots, 10$. It is easy to check that this linear system has a unique feasible solution; $a_2 = -1$, $a_3 = 7/12$ and $a_4 = 1/6$. This gives $\tilde{p}(2) = 1$, $\tilde{p}(5) = 0$ and $\tilde{p}(8) = 1$ implying that $p(x) = 1$ for every x with $|x| = 2$ or 8, and $p(x) = 0$ for every x with $|x| = 5$.

Now we consider a multilinear polynomial p' on $\{x_1, \ldots, x_8\}$ obtained from p by fixing $x_9 = x_{10} = 0$. We have that $p'(x) = 0$ for every x with $|x| = 0$ or 5, and $p'(x) = 1$ for every x with $|x| = 1$, 2 or 8 because p is so. Since we have 5 fixed points for a degree 4 polynomial $\tilde{p'}$, it is uniquely determined to

$$\tilde{p'}(k) \;=\; k - \binom{k}{2} + \frac{7}{12}\binom{k}{3} - \frac{1}{6}\binom{k}{4}. \tag{9}$$

Since the coefficient of every degree 3 monomial of p' is 0 or 1, Fact 5 (i) implies that the coefficient $7/12$ of the term $\binom{k}{3}$ in Eq.(9) must be a multiple of $1/\binom{10}{3} = 1/56$, a contradiction. □

We now proceed to the case of degree 5. Since the minimum value of n that satisfies $\log_n 5 < \log_6 3$ is 14, we seek a function on 14 variables. We have tried to solve IP (7) for $(n, k) = (14, 5)$ by the IP solver in GLPK package [35], but it never came back. So far, we could only obtain a partial result like the following:

Definition 4. *We say that a multilinear polynomial p is* bounded *if the coefficient of a monomial t in p is* 0 *or* 1 *whenever the degree of t is odd, and is* 0 *or* -1 *whenever the degree is even.*

Note that any polynomial obtained by a recursive use of E_2 or F_3 are bounded.

Fact 6. *There are no Boolean functions f on* 14 *variables such that (i) f is fully sensitive at the origin, and (ii) f can be represented by a bounded polynomial of degree* 5.

The proof is by checking the infeasiblity of IP (7) with additional constraints that $x_S \in \{0, 1\}$ when $|S|$ is odd and $x_S \in \{-1, 0\}$ when $|S|$ is even. We also use the result that the maximum number of edges in a $\mathcal{K}_4^-$- free 3-graph on 14 vertices is 126 that were recently shown by Markström and Talbot [36] using a computer verification to reduce the search space.

Through the experiments, we now feel that the following problem, which is easier than Problem 4, is already very hard.

Problem 5. *Is there a fully sensitive function whose degree is smaller than the degree of function obtained by the recursion of F_3?*

4.2 Average Density of Sign-Representing Polynomials

The final topic in this article is on the "density" of sign-representing polynomials of Boolean functions. Recently, we found a new method for obtaining an improved bound on the average density of Boolean functions by computer calculations [1].

	Lower Bound	Upper Bound
All Functions	$(0.11)2^n$	$(0.75)2^n$
Almost All Functions	$(0.11)2^n$	$(0.617)2^n$

Table 1: The known bounds on the PTF density of Boolean functions on $\{1,-1\}^n$. The right bottom is shown here.

In most of this section, we use $\{1,-1\}$ to represent Boolean values. The false or 0 is represented by 1, and the true or 1 is represented by -1. Let $f : \{1,-1\}^n \to \{1,-1\}$ be a Boolean function on n variables and let $p : \mathbb{R}^n \to \mathbb{R}$ be a real polynomial. Recall that p sign-represents f if $\text{sgn}(p(x)) = f(x)$ for all $x \in \{1,-1\}^n$ (i.e., $\text{sgn}(p(x)) > 0$ if $f(x) = 1$ and $\text{sgn}(p(x)) < 0$ if $f(x) = -1$). The expressive power of such a representation has been extensively investigated especially in complexity theory and in learning theory (see e.g., [37, 46, 40, 50, 51] and the references therein). The PTF *density* of a given Boolean function f is the minimum number of monomials with non-zero coefficient in a polynomial that sign-represents f. Note that the PTF density is depending on the choice of the domain of functions. In this section, we exclusively consider the case $\{1,-1\}^n$.

It is classically known that every Boolean function on n variables can be sign-represented by a polynomial with 2^n monomials. However, in general, less monomials are enough. For example, it is easy exercise to show that every two-variable Boolean function can be sign-represented by a polynomial with at most three monomials, not four.

In spite of a long history of investigations, there still is a large gap between the upper and lower bounds on the worst/average PTF density of Boolean functions. For the lower bounds, Saks [46, Theorem 2.27] noted that the result of Cover [14] implies that almost all Boolean functions on n variables have PTF density at least $(0.11)2^n$. To this date, this is the best known lower bound on the PTF density even for the worst case.

Recently, Oztop [41] (see also [42]) gave an elegant proof of the result that the PTF density of every n-variable function is at most $(0.75)2^n$. This improves the previously known bound of $(1 - \frac{1}{O(n)})2^n$ by O'Donnell and Servedio [40] and is the first result saying that a constant ratio, namely, 3/4, of all monomials are always enough to represent a Boolean function.

In this section, we briefly sketch the Oztop's method and see that a generalization of this method can yield a better bound if we consider the *average* case. Intuitively, we show that the problem of upper bounding the average density of n-variable Boolean functions can be reduced to a problem of computing (some modified version of) average density of k-variable Boolean functions for small k. Interestingly, an upper bound is improved just by increasing the computational effort. The best bound we have obtained so far is $(0.617)2^n$ and it is quite conceivable that this bound will further be improved. The known bounds on the PTF density of Boolean functions are summarized in Table 1.

4.2.1 Basics

Let p be a real valued polynomial that sign-represents a Boolean function $f : \{1,-1\}^n \to \{1,-1\}$. Since $x^2 = 1$ for $x \in \{1,-1\}$, we can assume without loss of generality that p is a linear combination of all 2^n multilinear monomials over $x_1, \ldots, x_n$. The *support* of p is a set of monomials with non-zero coefficient in p and the *density* of p is the size of the support of p. For a Boolean function f, the PTF density of f is the smallest density of a polynomial that sign-represents f.

It is very useful to writing this in vector notations. We follow the notation by Oztop [41, 42].

For $n \geq 1$, let $\mathbf{D}^n$ be a Hadamard matrix of order 2^n defined as

$$\mathbf{D}^1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \mathbf{D}^n = \begin{pmatrix} \mathbf{D}^{n-1} & \mathbf{D}^{n-1} \\ \mathbf{D}^{n-1} & -\mathbf{D}^{n-1} \end{pmatrix} (\text{for } n \geq 2).$$

The well known identities $\mathbf{D}^n\mathbf{D}^n = 2^n\mathbf{I}$ and $(\mathbf{D}^n)^{-1} = 2^{-n}\mathbf{D}^n$ are very useful. Each column of $\mathbf{D}^n$ is indexed by a monomial in $\mathcal{M}^n$ in the ordering of $1, x_1, x_2, x_2x_1, x_3, x_3x_1, x_3x_2, x_3x_2x_1, \ldots, x_nx_{n-1}\ldots x_1$. For a polynomial $p = \sum_{i=1}^{2^n} a_im_i$ where $m_i = \prod_{j \in S_i} x_j$ with $S_i \subseteq \{1, \ldots, n\}$, the column vector $\mathbf{a} = (a_1, \ldots, a_{2^n})^T$ is called the *coefficient vector* of p where we use the same ordering for monomials as above.

Then the column vector $\mathbf{D}^n\mathbf{a}$ represents the values of $p(x)$ where the assignments to $(x_n, x_{n-1}, \ldots, x_1)$ are ordered as $00\ldots00, 00\cdots01, 00\cdots10, 00\cdots11, \ldots, 11\cdots11$ (where 0's represent 1 and 1's represent -1). For a Boolean function f on n variables, let $\mathbf{f}$ denote the column vector of length 2^n whose elements are the values of $f(x)$ for all x. We call $\mathbf{f}$ as the *vector representation* of f.

In this notation, p sign-represents f iff $\mathbf{YD}^n\mathbf{a} > \mathbf{0}$, where $\mathbf{Y} = \text{diag}(\mathbf{f})$. If this is the case, we are allowed to say that $\mathbf{a}$ sign-represents f. The density of p is the number of non-zero elements of $\mathbf{a}$.

Below we briefly sketch the proof of the $(0.75)2^n$ upper bound on the PTF density of *every n*-variable Boolean function by Oztop [41, 42].

The following simple fact is extremely useful.

Fact 7. *([42, Theorem 1]) Let* $\mathbf{f}$ *be the vector representation of a Boolean function on n variables. The set of solutions of the inequality* $\text{diag}(\mathbf{f})\mathbf{D}^n\mathbf{a} > \mathbf{0}$ *is all positive linear combinations of the columns of* $\mathbf{D}^n\text{diag}(\mathbf{f})$.

Proof. $\text{diag}(\mathbf{f})\mathbf{D}^n\mathbf{a} > \mathbf{0}$ iff $\exists\mathbf{k} > \mathbf{0}[\text{diag}(\mathbf{f})\mathbf{D}^n\mathbf{a} = \mathbf{k}]$ iff $\exists\mathbf{k} > \mathbf{0}[\mathbf{a} = \frac{1}{2^n}\mathbf{D}^n\text{diag}(\mathbf{f})\mathbf{k}]$. □

Theorem 4. *[41] For any Boolean function on n variables, there exists a sign-representing polynomial with at most $2^n - 2^n/4$ monomials.*

Proof. (sketch) For an n-variable Boolean function f, let $\mathbf{a}$ denote the coefficient vector of a polynomial that sign-represents f. Let $\mathbf{f}$ denote the vector representation of f. We partition $\mathbf{a}$ and $\mathbf{f}$ into $\mathbf{a}_0$, $\mathbf{a}_1$ and $\mathbf{f}_0$, $\mathbf{f}_1$ of equal length, respectively. We can write as

$$\mathrm{diag}\begin{pmatrix} \mathbf{f}_0 \\ \mathbf{f}_1 \end{pmatrix}\begin{pmatrix} \mathbf{D}^{n-1} & \mathbf{D}^{n-1} \\ \mathbf{D}^{n-1} & -\mathbf{D}^{n-1} \end{pmatrix}\begin{pmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \end{pmatrix} > \mathbf{0}.$$

This is equivalent to

$$\begin{aligned} \mathrm{diag}(\mathbf{f}_0)\mathbf{D}^{n-1}(\mathbf{a}_0 + \mathbf{a}_1) &> \mathbf{0}, \\ \mathrm{diag}(\mathbf{f}_1)\mathbf{D}^{n-1}(\mathbf{a}_0 - \mathbf{a}_1) &> \mathbf{0}. \end{aligned} \tag{10}$$

By Fact 7, this is equivalent to

$$\begin{aligned} (\mathbf{a}_0 + \mathbf{a}_1)^{\mathrm{T}} &= 2\mathbf{k}_0\mathbf{Y}_0, \\ (\mathbf{a}_0 - \mathbf{a}_1)^{\mathrm{T}} &= 2\mathbf{k}_1\mathbf{Y}_1. \end{aligned} \tag{11}$$

for some row vectors $\mathbf{k}_0 > \mathbf{0}$ and $\mathbf{k}_1 > \mathbf{0}$, where $\mathbf{Y}_i$ ($i = 0, 1$) denotes $2^{n-1} \times 2^{n-1}$ matrix $\mathrm{diag}(\mathbf{f}_i)\mathbf{D}^{n-1}$.

Let $\mathbf{Z}_0$ (resp, $\mathbf{Z}_1$) be a matrix consisting of all rows of $\mathbf{Y}_0$ indexed by $x \in \{0,1\}^{n-1}$ such that $f_0(x) = f_1(x)$ ($f_0(x) \neq f_1(x)$, resp). Then Eq. (11) can be written as

$$\begin{aligned} (\mathbf{a}_0 + \mathbf{a}_1)^{\mathrm{T}} &= 2\mathbf{k}_{0,0}\mathbf{Z}_0 + 2\mathbf{k}_{0,1}\mathbf{Z}_1, \\ (\mathbf{a}_0 - \mathbf{a}_1)^{\mathrm{T}} &= 2\mathbf{k}_{1,0}\mathbf{Z}_0 - 2\mathbf{k}_{1,1}\mathbf{Z}_1. \end{aligned} \tag{12}$$

where $\mathbf{k}_{i,j} > \mathbf{0}$ ($i, j = 0, 1$) is a suitable partition of $\mathbf{k}_i$ ($i = 0, 1$). By solving this for $\mathbf{a}_0$ and $\mathbf{a}_1$, we have

$$\begin{aligned} \mathbf{a}_0^{\mathrm{T}} &= (\mathbf{k}_{0,0} + \mathbf{k}_{1,0})\mathbf{Z}_0 + (\mathbf{k}_{0,1} - \mathbf{k}_{1,1})\mathbf{Z}_1, \\ \mathbf{a}_1^{\mathrm{T}} &= (\mathbf{k}_{0,0} - \mathbf{k}_{1,0})\mathbf{Z}_0 + (\mathbf{k}_{0,1} + \mathbf{k}_{1,1})\mathbf{Z}_1. \end{aligned}$$

Let z_0 and z_1 denote the number of rows in $\mathbf{Z}_0$ and $\mathbf{Z}_1$, respectively. Note that $z_0 + z_1 = 2^{n-1}$. If $z_1 \geq 2^{n-2}$, then for any $\mathbf{k}_{0,0}$ and $\mathbf{k}_{1,0}$, we can zero z_1 components of $\mathbf{a}_0$ by an appropriate setting of $\mathbf{k}_{0,1}$ and $\mathbf{k}_{1,1}$ since $\mathbf{Z}_1$ has full rank (this is because every rows in $\mathbf{D}^{n-1}$ are linearly independent). If $z_1 < 2^{n-2}$, which implies $z_0 > 2^{n-2}$, then for any $\mathbf{k}_{0,1}$ and $\mathbf{k}_{1,1}$, we can zero z_0 components of $\mathbf{a}_1$ by an appropriate setting of $\mathbf{k}_{0,0}$ and $\mathbf{k}_{1,0}$. This means that a polynomial given by coefficient $\mathbf{a} = (\mathbf{a}_0, \mathbf{a}_1)$ sign-represents f and has at most $2^n - 2^n/4$ non-zero elements. □

As we see, this proof is based on a decomposition of f into two subfunctions $f|_{x_n=1}$ and $f|_{x_n=-1}$. It is quite natural to ask what happens if we decompose f by fixing two or more variables. We below see that this gives an improved bound for the *average* case.

4.2.2 Finer Decompositions yield Better Bounds

Let's see what happens if we consider two-variable decompositions in the proof of Theorem 4 instead of one. We partition $\mathbf{f}$ into four parts of equal length $\mathbf{f}_{00}$, $\mathbf{f}_{01}$, $\mathbf{f}_{10}$ and $\mathbf{f}_{11}$. Note that $\mathbf{f}_{i,j}$ is the vector representation of the function $f|_{x_n=i,x_{n-1}=j}$ (here we consider the input is $\{0,1\}$ instead of $\{+1,-1\}$), which we will denote $f_{i,j}$. Similarly the coefficient vector $\mathbf{a}$ is partitioned into four vectors of equal length $\mathbf{a}_{00}$, $\mathbf{a}_{01}$, $\mathbf{a}_{10}$ and $\mathbf{a}_{11}$. In place of Eq. (12), we should introduce 2^3 submatrices $\mathbf{Z}_p$ for $p \in \{0,1\}^3$ these are defined according to the values of $(f_{00}(x), f_{01}(x), f_{10}(x), f_{11}(x))$. Then by a similar argument to the proof of Theorem 4, we can make zero using the freedom of $\mathbf{k}$'s. Note that in this case we can use another freedom to enlarge the number of zeros in $\mathbf{a}$. We can consider any ordering of the partitions of $\mathbf{a}$, e.g., $\mathbf{a}_{10} \to \mathbf{a}_{11} \to \mathbf{a}_{01} \to \mathbf{a}_{00}$.

By taking all of them into account, the number of zeros we can guarantee by this procedure is formulated as follows: Let $\mathcal{M}^k$ denote the all 2^k monomials on k variables and π be a mapping from $\{1,\ldots,2^k\}$ to $\mathcal{M}^k$. The mapping π represents the ordering of the monomials $\{\pi(1),\pi(2),\ldots,\pi(2^k)\}$. For a Boolean function f and an ordering of monomials π, the *freedom* of f with respect to π, denoted by $\mathsf{free}(f,\pi)$ is defined as the maximum t such that f can be sing-represented by a polynomial with monomials $\mathcal{M}^k - \{\pi(1),\pi(2),\ldots,\pi(t)\}$. For a positive integer k and an ordering π of $\mathcal{M}^k$, let $d(k,\pi)$ denote the average of $\mathsf{free}(f,\pi)/2^k$ over all k-variable Boolean functions f with $f(1,1,\ldots,1) = 1$ which we call the *average freedom* with respect to π. Note that the last condition ($f(1,1,\ldots,1) = 1$) can be removed without changing the value of $d(k,\pi)$ by symmetry.

The generalization of Theorem 4 can be stated as follows (see [1] for the proof).

Theorem 5. *[1] Let $\epsilon > 0$ an arbitrary constant. Let $k \geq 1$ be an integer and π be an ordering of $\mathcal{M}^k$. Then, there is a constant $c > 0$ (depending on ϵ and k) such that all but a 2^{-c2^n} fraction of n-variable Boolean functions have PTF density at most $(1 - d(k,\pi) + \epsilon)2^n$.*

It is a bit tedious but easy to verify (by hand) that $d(2,\pi) = 5/16$ for every π of $\mathcal{M}^2$; this gives $(0.688)2^n$ upper bounds on the average PTF density of n-variable functions. The computation of $d(k,\pi)$ for $k \geq 3$ is done by a computer (it takes double exponential time in k). Note that we can see whether a given k-variable function f has a sign-representing polynomial with support S by checking the feasibility of the linear system $\mathrm{diag}(\mathbf{f})\mathbf{D}^k\mathbf{a} > \mathbf{0}$ where $a_i = 0$ for every $i \notin S$, which is an easy task for any linear programming solver when k is small.

The computation of $d(3,\pi)$ for every possible π is quite feasible by a standard

PC. By using the GLPK package [35], we found that

$$\begin{aligned} d(3, \{1, x_1, x_2, x_1x_2, x_3, x_1x_3, x_2x_3, x_1x_2x_3\}) &= 316/(2^7 \times 8) = 0.3085\cdots, \\ d(3, \{1, x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1x_2x_3\}) &= 360/(2^7 \times 8) = 0.3515\cdots, \end{aligned}$$

and that all orderings of $\mathcal{M}^3$ are categorized into one of the above two. This gives the upper bound on the average PTF density of $(0.649)2^n$.

The computation of $d(4, \pi)$ for all π seems out of reach. However, the result for $k = 3$ inspires the ordering

$$\pi_4 = \{1, x_1, \ldots, x_4, x_1x_2, \ldots, x_3x_4, x_1x_2x_3, \ldots, x_2x_3x_4, x_1x_2x_3x_4\},$$

would be a good candidate. We compute $d(4, \pi_4)$ (again by a computer) to find

$$d(4, \pi_4) = 195804/(2^{15} \times 16) = 0.3734\cdots, \tag{13}$$

which gives the upper bound of $(0.627)2^n$. However, this is not the best.

We then computed the average freedom of 2000 randomly chosen orderings of $\mathcal{M}^4$, which took about two days on a standard PC. The best ordering we have found is

$$\begin{aligned} \pi'_4 = \{&1, x_1x_2x_3, x_1x_4, x_1x_3x_4, x_1x_2x_3x_4, x_3x_4, x_2x_4, x_3, x_1x_3, \\ &x_1, x_1x_2, x_2x_3x_4, x_2x_3, x_2, x_4\}, \end{aligned}$$

which has the average freedom of

$$d(4, \pi'_4) = 200964/(2^{15} \times 16) = 0.3833\cdots. \tag{14}$$

Note that 8 (out of 2000) orderings have the same value. This immediately gives the following upper bound, which is the best we have obtained so far.

Corollary 1. *[1] Almost all Boolean functions on n variables have PTF density at most* $(0.617)2^n$.

We provide verifiable data for Eqs. (13) and (14) on the web page [1]. These are the lists of polynomial representations of all 4-variable functions with maximum freedom with respect to the designated ordering. The correctness of Eqs. (13) and (14) can be verified by hand in a several weeks, or by a computer in a few seconds. At the time of writing this article, we don't know whether π'_4 is the best among all orderings of $\mathcal{M}^4$ or not.

Apparently, our method would yield a better upper bound if we have more computational resource (or more sophisticated algorithm for computing the average freedom). A random sampling experiment suggests that $1 - d(5, \pi_5)$ is around 0.598 where π_5 is an obvious extension of π_4. A natural question is:

Problem 6. *What is the best achievable bound obtained by this method?*

5 Concluding Remarks

In this article, we review several recent attempts to use computers in various ways to obtain concrete results for the problems in computational complexity. Some of them are (at least partially) succeeded and some of them are not. We strongly hope that these approaches inspire a new idea on how to attack the difficult problems, especially the lower bound problems, in computational complexity.

The last thing we mention is that the feasibility of checking the computer proofs. If a proof is "NP-type", i.e., a witness of an upper or lower bound is generated (like the one in Section 4.2), then it seems no problems. However, if a proof is "co-NP-type", i.e., the non-existence is verified by checking a huge number of cases (like the one in Section 3.3), this may be a problematic. One possible approach is to translate a computer proof into a formal proof. A Coq proof of the Four Color Theorem by Gonthier [19] or an ongoing "Flyspeck" project [20] for the Kepler Conjecture is a nice example. However this would need a huge amount of work in general, and we do not have a good solution to this meta-problem.

References

[1] K. Amano, New Upper Bounds on the Average PTF Density of Boolean Functions, *Manuscript*, 2010 (available at `http://www.cs.gunma-u.ac.jp/~amano/poly/index.html`).

[2] K. Amano and A. Maruoka. On the Complexity of Depth-2 Circuits with Threshold Gates, *Proc. of MFCS '05*, LNCS 3618, 107-118, 2005.

[3] K. Amano and A. Maruoka. Better Upper Bounds on the QOBDD Size of Integer Multiplication, *Disc. Appl. Math.*, 155(10): 1224–1232, 2007.

[4] K. Amano and J. Tarui. A Well-Mixed Function with Circuit Complexity $5n \pm o(n)$: Tightness of the Lachish-Raz-type Bounds, *Proc. of TAMC '08*, LNCS 4978, 342–350, 2008.

[5] A. Ambainis, Polynomial Degree vs. Quantum Query Complexity, *J. Comput. Sys. Sci.*, 72(2), 220–238, 2006 (Earlier version in FOCS '03).

[6] K. Appel and W. Haken, Every Planar Map is Four Colorable. Part I. Discharging, *Illinois J. Math.*, 21, 429–490, 1977.

[7] K. Appel, W. Haken and J. Koch, Every Planar Map is Four Colorable. Part II. Reducibility, *Illinois J. Math.*, 21, 491–567, 1977.

[8] H. Barnum, M.E. Saks and Mario Szegedy, Quantum query complexity and semidefinite programming. *Proc. of CCC '03*, 179–193, 2003.

[9] S. Basu, N. Bhatnagar, P. Gopalan and R.J. Lipton, Polynomials that Sign Represent Parity and Descartes Rule of Signs, *Computational Complexity*, 17(3), 377-406, 2008. (Earlier version in CCC '04)

[10] N. Blum, A Boolean Function Requiring 3n Network Size, *Theoret. Comput. Sci.*, 28, 337–345, 1984.

[11] B. Bollig, Integer Multiplication and the Complexity of Binary Decision Diagrams, *Bulletin of the EATCS*, 98, 78–106, 2009.

[12] R.E. Bryant, On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Applications to Integer Multiplication, *IEEE Trans. on Computers*, 40, 205–213, 1991.

[13] H. Buhrman and R. de Wolf, Complexity Measures and Decision Tree Complexity: A Survey, *Theoret. Comput. Sci.*, 288(1), 21–43, 2002.

[14] T. Cover, Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition, *IEEE Trans. Electronic Computers*, EC-14(3), 326–334, 1965.

[15] J. Forster, M. Krause, S. V. Lokam, R. Mubarakzjanov, N. Schmitt, H. Simon, Relations Between Communication Complexity, Linear Arrangements, and Computational Complexity. *Proc. of FSTTCS '01*, 171–182, 2001.

[16] P. Frankl and Füredi, An Exact Result for 3-Graphs, *Discrete Math.*, 50(2-3), 323–328, 1984.

[17] S.J. Friedman and K.J. Supowit, Finding the Optimal Variable Ordering for Binary Decision Diagrams, *IEEE Trans. Comput.*, 39, 710–713, 1990.

[18] H. Fukuhara, *personal communication*, 2010.

[19] G. Gonthier, Formal Proof – The Four-Color Theorem, *Notices of the AMS*, 55(11), 1382–1393, 2008.

[20] T.C. Hales, J. Harrison, S. McLaughlin, T. Nipkow, S. Obua and R. Zumkeller, A Revision of the Proof of the Kepler Conjecture, *Discrete Comput. Geom*, 44, 1–34, 2010.

[21] J. Håstad, The Shrinkage Exponent of de Morgan Formulas is 2, *SIAM J. Comput.*, 27(1), 48–64, 1998.

[22] P. Høyer, T. Lee, R. Sp̌alek Tight Adversary Bounds for Composite Functions, `quant-ph/0509067`, 2005.

[23] P. Høyer, T. Lee, R. Sp̌alek Negative Weights Make Adversaries Stronger, *Proc. of STOC '07*, 526–535, 2007.

[24] P. Hrubes, S. Jukna, A. Kulikov and P. Pudlak, On Convex Complexity Measures, *Theoret. Comput. Sci.*, 411, 1842–1854, 2010.

[25] K. Iwama and H. Morizumi, An Explicit Lower Bound of $5n - o(n)$ for Boolean Circuits, *Proc. of MFCS '02*, 353–364, 2002.

[26] K. Iwama, O. Lachish, H. Morizumi and R. Raz, An Explicit Lower Bound of $5n - o(n)$ for Boolean Circuits, *Manuscript*, (combined version of [25] and [34]), 2005.

[27] V.M. Khrapchenko, A Method of Determining Lower Bounds for the Complexity of Π-schemes, *Mathematicheskie Zametki*, 10(1), 83–92, 1971 (in Russian); English translation in *Mathematical Notes*, 10(1), 474–479, 1971.

[28] A. A. Kojevnikov, A. S. Kulikov and G. N. Yaroslavtsev, Finding Efficient Circuits Using SAT-solvers, *Proc. of SAT '09*, LNCS 5584, 32–44, 2009.

[29] E. Kushilevits and E. Weinreb, On the Complexity of Communication Complexity, *Proc. of STOC '09*, 465–474, 2009.

[30] D. Knuth, The Art of Computer Programming: Volume 4, Fascicle 0: Introduction to Combinatorial Algorithms and Boolean Functions, Addison-Wesley, 2008.

[31] D. Knuth, The Art of Computer Programming: Volume 4, Fascicle 1: Bitwise Tricks & Techniques; Binary Decision Diagrams, Addison-Wesley, 2009.

[32] S. Laplante, T. Lee and M. Szegedy, The Quantum Adversary Method and Classical Formula Size Lower Bounds, *Computational Complexity*, 15, 163–196, 2006. (Earlier version in CCC' 05).

[33] S. Laplante, F. Magniez, Lower Bounds for Randomized and Quantum Query Complexity using Kolmogorov Arguments, *Proc. of CCC '04*, 294–304, 2004.

[34] O. Lachish and R. Raz, Explicit Lower Bound of $4.5n - o(n)$ for Boolean Circuits, *Proc. of STOC '01*, 399–408, 2001.

[35] A. Makhorin, The GLPK (GNU Linear Programming Kit) Package, available at `http://www.gnu.org/software/glpk/`.

[36] K. Markström and J. Talbot, On the Density of 2-Colourable 3-Graphs in which any Four Points Span at Most Two Edges, *J. Combin. Designs*, 18(2), 105–114, 2009.

[37] M. Minsky and S. Papert, Perceptrons, MIT Press, Cambridge, 1988 (First edition appeared in 1968)

[38] N. Nisan and M. Szegedy, On the Degree of Boolean Functions as Real Polynomials, *Computational Complexity*, 4, 301–313, 1994.

[39] N. Nisan and A. Wigderson, On Rank vs. Communication Complexity, *Combinatorica*, 15(4), 557–565, 1995. (Earlier version in FOCS '94)

[40] R. O'Donnell, R. Servedio, Extremal Properties of Polynomial Threshold Functions, *J. Comput. Syst. Sci.*, 74(3), 298-312, 2008. (Earlier version in CCC'03)

[41] E. Oztop, An Upper Bound on the Minimum Number of Monomials Required to Separate Dichotomies of $\{-1, 1\}^n$, Neural Computation, 18(12), 3119-3138, 2006.

[42] E. Oztop, Sign-representation of Boolean Functions using a Small Number of Monomials, Neural Networks, 22(7), 938-948, 2009.

[43] A. Razborov, Flag Algebras, *J. Symbolic Logic*, 72(4), 1239–1282, 2007.

[44] A. Razborov, On 3-Hypergraphs with Forbidden 4-Vertex Configurations, *Manuscript*, 2008.

[45] R. Robertson, D.P. Sanders, P.D. Seymour and R. Thomas, The Four Colour Theorem, *J. Combin. Theory.*, Ser. B, 70, 2–44, 1997.

[46] M.E. Saks, Slicing the Hypercubes, *Surveys in Combinatorics*, Cambridge University Press, 211–255, 1993.

[47] M. Sauerhoff, An Asymptotically Optimal Lower Bound on the OBDD Size of the Middle Bit of Multiplication for the Pairwise Ascending Variable Order, *Disc. Appl. Math.*, 158(11), 1195–1204, 2010.

[48] P. Savický and S. Žák, A Large Lower Bound for 1-Branching Programs, *ECCC TR96-036 Rev.01*, 1996.

[49] A.A. Sherstov, The Unbounded-Error Communication Complexity of Symmetric Functions, *Proc. of FOCS '08*, 384–393, 2008.

[50] A.A. Sherstov, The Intersection of Two Halfspaces has High Threshold Degree, *Proc. of FOCS '09*, 343–362, 2009.

[51] A.A. Sherstov, Optimal Bounds for Sign-representing the Intersection of Two Halfspaces by Polynomials, *Proc. of STOC '10*, 2010.

[52] R.Špalek and M. Szegedy, All Quantum Adversary Methods are Equivalent, *Proc. of ICALP '05*, 1299-1311, 2005.

[53] J. Tarui, Smallest Formulas for the Parity of 2^k Variables are Essentially Unique, *Theoret. Comput. Sci.*, (in press), 2010. (Earlier version in COCOON '08)

[54] R. Williams, Applying Practice to Theory, *ACM SIGACT News*, 39(4): 37–52, 2008.

[55] P. Woelfel, Bounds on the OBDD-size of Integer Multiplication via Universal Hashing. *J. Comput. Sys. Sci.*, 71(4), 520–534, 2005.

[56] S. Zhang, On the Power of Ambainis's Lower Bounds, *Theoret. Comput. Sci.*, 339(2-3), 241–256, 2005.