Circuit Size Lower Bounds and #SAT Upper Bounds Through a General Framework *

Alexander Golovnev[1,2], Alexander S. Kulikov[2], Alexander V. Smal[3], and Suguru Tamaki[4]

1 **New York University, USA and St. Petersburg Department of Steklov Institute of Mathematics of the Russian Academy of Sciences, Russia**
alex.golovnev@gmail.com
2 **St. Petersburg Department of Steklov Institute of Mathematics of the Russian Academy of Sciences, Russia**
kulikov@logic.pdmi.ras.ru
3 **St. Petersburg Department of Steklov Institute of Mathematics of the Russian Academy of Sciences, Russia**
smal@logic.pdmi.ras.ru
4 **Kyoto University, Japan**
tamak@kuis.kyoto-u.ac.jp

Abstract

Most of the known *lower bounds* for binary Boolean circuits with unrestricted depth are proved by the gate elimination method. The most efficient known *algorithms* for the #SAT problem on binary Boolean circuits use similar case analyses to the ones in gate elimination. Chen and Kabanets recently showed that the known case analyses can also be used to prove *average case circuit lower bounds*, that is, lower bounds on the size of approximations of an explicit function.

In this paper, we provide a general framework for proving worst/average case lower bounds for circuits and upper bounds for #SAT that is built on ideas of Chen and Kabanets. A proof in such a framework goes as follows. One starts by fixing three parameters: a class of circuits, a circuit complexity measure, and a set of allowed substitutions. The main ingredient of a proof goes as follows: by going through a number of cases, one shows that for any circuit from the given class, one can find an allowed substitution such that the given measure of the circuit reduces by a sufficient amount. This case analysis immediately implies an upper bound for #SAT. To obtain worst/average case circuit complexity lower bounds one needs to present an explicit construction of a function that is a disperser/extractor for the class of sources defined by the set of substitutions under consideration.

We show that many known proofs (of circuit size lower bounds and upper bounds for #SAT) fall into this framework. Using this framework, we prove the following new bounds: average case lower bounds of $3.24n$ and $2.59n$ for circuits over U_2 and B_2, respectively (though the lower bound for the basis B_2 is given for a quadratic disperser whose explicit construction is not currently known), and faster than 2^n #SAT-algorithms for circuits over U_2 and B_2 of size at most $3.24n$ and $2.99n$, respectively. Here by B_2 we mean the set of all bivariate Boolean functions, and by U_2 the set of all bivariate Boolean functions except for parity and its complement.

1998 ACM Subject Classification F.1.1 Models of Computation

* The research presented is Section 3.1 is supported in part by MEXT KAKENHI (24106003); JSPS KAKENHI (26330011, 16H02782); the John Mung Advanced Program of Kyoto University. The research presented in Section 3.2 is partially supported by NSF grant 1319051. The research presented in Sections 4–5 is supported by Russian Science Foundation (project 16-11-10123). Part of the work was performed while Suguru Tamaki was at Department of Computer Science and Engineering, University of California, San Diego and the Simons Institute for the Theory of Computing, Berkeley.

41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016).
Editors: Piotr Faliszewski, Anca Muscholl, and Rolf Niedermeier; Article No. 45; pp. 45:1–45:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Keywords and phrases circuit complexity, lower bounds, exponential time algorithms, satisfiability

Digital Object Identifier 10.4230/LIPIcs.MFCS.2016.45

1 Introduction

1.1 Background

In this paper, we study binary Boolean circuits with no restriction on the depth. This is a natural model for computing Boolean functions that can be viewed as a simple program where each instruction is just a binary Boolean operation. Shannon [56] showed that for almost all Boolean functions of n variables the size of a smallest circuit (equivalently, the minimal number of instructions) computing this function is $\Omega(2^n/n)$. The proof is based on a counting argument (the number 2^{2^n} of all functions of n variables is larger than the number of circuits of size $o(2^n/n)$) and, for this reason, does not give an explicit function of high circuit complexity. By saying "explicit" one usually means a function from NP. Showing a superpolynomial lower bound for an explicit function would imply P $\neq$ NP. However, despite of many efforts [53, 47, 57, 8, 20, 26, 66, 3], currently we have only small linear lower bounds: $(3 + 1/86)n$ for the full binary basis B_2 consisting of all binary Boolean functions [25] and $5n - o(n)$ for the basis U_2 consisting of all binary Boolean functions except for parity and its complement [38, 32].

Going to larger complexity classes, it is known that the classes MA/1 [50], O_2^p [10], and $\mathrm{P}^{\mathrm{prMA}}$ [11] require circuits of superlinear size and the class MAEXP [9] has superpolynomial circuit complexity. Proving a superlinear lower bound on the circuit complexity of E^{NP} remains to be a major open problem.

Recently, Williams [60, 64] presented the following approach to prove circuit size lower bounds against E^{NP} or NE using SAT-algorithms: a super-polynomially faster than 2^n algorithm for the circuit satisfiability problem of a "reasonable" circuit class $\mathcal{C}$ implies either $\mathrm{E}^{\mathrm{NP}} \not\subseteq \mathcal{C}$ or NE $\not\subseteq \mathcal{C}$, depending on $\mathcal{C}$ and the running time of the algorithm. The approach has been strengthened and simplified by subsequent work [59, 61, 63, 7, 33], see also excellent surveys [52, 45, 62] on this topic.

Williams' result inspired lots of work on satisfiability algorithms for various circuit classes [30, 63, 15, 2, 1, 43, 16, 58]. In addition to satisfiability algorithms, several papers [51, 29, 4, 54, 14, 12, 17, 49] also obtained average-case lower bounds (also known as correlation bounds, see [35, 36, 28]) by investigating the analysis of algorithms instead of just applying Williams' result that yields worst-case lower bounds. In particular, Chen and Kabanets [13] presented algorithms that count the number of satisfying assignments of circuits over U_2 and B_2 and run in time exponentially faster than 2^n if input instances have at most $2.99n$ and $2.49n$ gates, respectively (improving also the previously best known #SAT-algorithm by Nurk [44]). At the same time, they showed that $2.99n$ sized circuits over U_2 and $2.49n$ sized circuits over B_2 have exponentially small correlations with the parity function and affine extractors having "good" parameters, respectively.

To prove a lower bound of ζn on the circuit size, one usually shows that for any circuit there is a substitution $x_i \leftarrow f$ eliminating at least ζ gates from the circuit. For example, to prove a lower bound of $3n - 3$ on the circuit size over U_2 of the parity function, Schnorr [53] shows how to make a bit-fixing substitution (i.e., $f = c$ for $c \in \{0, 1\}$) eliminating at least 3 gates from any U_2-circuit. Demenkov and Kulikov [20] prove a lower bound of $3n - o(n)$ on the circuit size over B_2 of an affine disperser by showing that for any B_2-circuit there is an affine substitution ($f = \oplus_{j\in J} x_j \oplus c$) eliminating at least three gates from the circuit. Chen and Kabanets proved new average case lower bounds and #SAT upper bounds by analyzing what happens in the complementary branch $x_i \leftarrow f \oplus 1$ of proofs by Schnorr and by Demenkov and Kulikov.

1.2 Our Techniques and Results

The main qualitative contribution of this paper is a general framework for proving circuit worst/average case lower bounds and #SAT upper bounds. This framework is separated into conceptual and technical parts. The conceptual part is a proof that for a given circuit complexity measure and a set of allowed substitutions, for any circuit, there is a substitution that reduces the complexity of the circuit by a sufficient amount. This is usually shown by analyzing the structure of the top of a circuit. The technical part is a set of lemmas that allows us to derive worst/average case circuit size lower bounds and #SAT upper bounds as one-line corollaries from the corresponding conceptual part. The technical part can be used in a black-box way: given a proof that reduces the complexity measure of a circuit (conceptual part), the technical part implies circuit lower bounds and #SAT upper bounds. For example, by plugging in the proofs by Schnorr and by Demenkov and Kulikov, one immediately gets the bounds given by Chen and Kabanets. We also give new proofs that lead to the quantitatively better results. The main quantitative contribution of the paper is the following new bounds which are currently the strongest known bounds:

- average case lower bounds of $3.24n$ and $2.59n$ for circuits over U_2 and B_2 (though the lower bound for the basis B_2 is given for a quadratic disperser whose explicit construction is not currently known), respectively, improving upon the bounds of $2.99n$ and $2.49n$ [13];
- faster than 2^n #SAT-algorithms for circuits over U_2 and B_2 of size at most $3.24n$ and $2.99n$, respectively, improving upon the bounds of $2.99n$ and $2.49n$ [13].

These bounds are obtained by using non-standard circuit complexity measures and sets of substitutions. We also show that obtaining non-linear lower bounds through a weak version of this framework is unlikely as it would violate the Exponential Time Hypothesis [31] that states the following: The satisfiability problem of 3-CNF formulas with n variables cannot be solved in time $2^{o(n)}$. ETH is widely used as a hardness assumption to prove the optimality of many algorithms, see, e.g., [41, 42].

1.3 Framework

We prove circuit lower bounds (both in the worst case and in the average case) and upper bounds for #SAT using the following four step framework.

Initial setting We start by specifying the three main parameters: a class of circuits $\mathcal{C}$, a set $\mathcal{S}$ of allowed substitutions, and a circuit complexity measure μ. A set of allowed substitutions naturally defines a class of "sources". For the circuit lower bounds we consider functions that are non-constant (dispersers) or close to uniform (extractors) on corresponding sets of sources. In this paper we focus on the following four sets of substitutions where each set extends the previous one:

1. Bit fixing substitutions, $\{x_i \leftarrow c\}$: substitute variables by constants.
2. Projections, $\{x_i \leftarrow c, x_i \leftarrow x_j \oplus c\}$: substitute variables by constants and other variables and their negations.

3. Affine substitutions, $\{x_i \leftarrow \bigoplus_{j \in J} x_j \oplus c\}$: substitute variables by affine functions of other variables.
4. Quadratic substitutions, $\{x_i \leftarrow p\colon \deg(p) \leq 2\}$: substitute variables by degree two polynomials of other variables.

Case analysis We then prove the main technical result stating that for any circuit from the class $\mathcal{C}$ there exists (and can be constructed efficiently) an allowed substitution $x_i \leftarrow f \in \mathcal{S}$ such that the measure μ is reduced by a sufficient amount under both substitutions $x_i \leftarrow f$ and $x_i \leftarrow f \oplus 1$.

#SAT upper bounds As an immediate consequence, we obtain an upper bound on the running time of an algorithm solving #SAT for circuits from $\mathcal{C}$. The corresponding algorithm takes as input a circuit, branches into two cases $x_i \leftarrow f$ and $x_i \leftarrow f \oplus 1$, and proceeds recursively. When applying a substitution $x_i \leftarrow f \oplus c$, it replaces all occurrences of x_i by a subcircuit computing $f \oplus c$. The case analysis provides an upper bound on the size of the resulting recursion tree.

Circuit size lower bounds Then, by taking a function that survives under sufficiently many allowed substitutions, we obtain lower bounds on the average case and worst case circuit complexity of the function. Below, we describe such functions, i.e., dispersers and extractors for the classes of sources under consideration.

1. The class of bit fixing substitutions generates the class of *bit-fixing sources* [18]. Extractors for bit-fixing sources find many applications in cryptography (see [22] for an excellent survey of the topic). The standard function that is a good disperser and extractor for such sources is the parity function $x_1 \oplus \cdots \oplus x_n$.
2. Projections define the class of *projection sources* [46]. Dispersers for projections are used to prove lower bounds for depth-three circuits [46]. It is shown [46] that a binary BCH code with appropriate parameters is a disperser for $n-o(n)$ substitutions. See [48] for an example of extractor with good parameters for projection sources.
3. Affine substitutions give rise to the class of *affine sources*. Dispersers for affine sources find applications in circuit lower bounds [19, 20, 25]. There are several known constructions of dispersers [6, 55] and extractors [65, 39, 5, 40] that are resistant to $n - o(n)$ substitutions.
4. The class of quadratic substitutions generates a special case of *polynomial sources* [24, 5] and *quadratic varieties sources* [23]. An explicit construction of disperser for quadratic varieties sources would imply new circuit lower bounds [26]. Although an explicit construction of a function resistant to sufficiently many quadratic substitutions is not currently known, it is easy to show that a random function is resistant to any $n - o(n)$ quadratic substitutions.

Due to the page limit of this extended abstract, we have to omit many proofs, which can be found in the full version [27].

2 Preliminaries

2.1 Boolean functions

We denote by B_n the set of all n-variate Boolean functions and define $U_2 = B_2 \setminus \{\oplus, \equiv\}$ as the set of all *binary* Boolean functions except for parity and its complement.

The set of all sixteen binary Boolean functions $f(x, y) \in B_2$ can be classified as follows: 1) two constant functions: 0 and 1; we also call them *trivial*; 2) four functions that depend essentially on one of the arguments only: x, $x \oplus 1$, y, $y \oplus 1$; we call them *degenerate*; 3) eight *and-type* functions: $(x \oplus a) \cdot (y \oplus b) \oplus c$ where $a, b, c \in \{0, 1\}$; 4) two *xor-type* functions: $x \oplus y \oplus a$, where $a \in \{0, 1\}$.

Hence U_2 consists of all binary functions except for xor-type functions. An important property of binary and-type functions $(x \oplus a) \cdot (y \oplus b) \oplus c$, useful for case analyses, is the following: one can turn this function into a constant c by assigning $x \leftarrow a$ or $y \leftarrow b$.

2.2 Dispersers and Extractors

Let $x_1, \ldots, x_n$ be Boolean variables, and $f \in B_{n-1}$ be a function of $n-1$ variables. We say that $x_i \leftarrow f(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ is a substitution to the variable x_i.

Let $g \in B_n$ be a function, then the *restriction* of g under the substitution f is a function $h = (g|x_i \leftarrow f)$ of $n-1$ variables, such that $h(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = g(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$, where $x_i = f(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$. Similarly, if $K \subseteq \{0, 1\}^n$ is a subset of the Boolean cube, then the *restriction* of K under this substitution is $K' = (K|x_i \leftarrow f)$, such that $(x_1, \ldots, x_n) \in K'$ if and only if $(x_1, \ldots, x_n) \in K$ and $x_i = f(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$.

For a family of functions $\mathcal{F} = \{f : \{0, 1\}^* \to \{0, 1\}\}$ we define a set of corresponding substitutions $\mathcal{S}(\mathcal{F})$ that contains the following substitutions: for every $1 \leq i \leq n, c \in \{0, 1\}, f \in \mathcal{F}$, $\mathcal{S}$ contains the substitution $x_i \leftarrow f(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \oplus c$.

Let $\mathcal{S}$ be a set of substitutions. We say that a set $K \subseteq \{0, 1\}^n$ is an $(\mathcal{S}, n, r)$-*source*[1] if it can be obtained from $\{0, 1\}^n$ by applying at most r substitutions from $\mathcal{S}$.

A function $f \in B_n$ is called an $(\mathcal{S}, n, r)$-*disperser*[2] if it is not constant on every $(\mathcal{S}, n, r)$-source. A function $f \in B_n$ is called an $(\mathcal{S}, n, r, \varepsilon)$-*extractor* if $|\Pr_{x \leftarrow K}[f(x) = 1] - 1/2| \leq \varepsilon$ for every $(\mathcal{S}, n, r)$-source K.

2.3 Circuits

A circuit over the basis $\Omega \subseteq B_2$ is a directed acyclic graph with the following properties: 1) the indegree of each node is either zero or two; 2) each node of zero indegree is labeled by a variable and is called an *input* or an *input gate*; 3) each node of indegree two is labeled with a binary Boolean function from Ω called an *operation* of this gate; the node itself is called an *internal gate* or just a *gate*; 4) there is a unique node of outdegree zero and it is called an *output*. Such a circuit computes in a natural way a function from B_n, where n is the number of input gates of the circuit. In this paper, we consider circuits over the bases $\Omega = B_2$ and $\Omega = U_2$.

An *xor-gate* (*and-gate*) is a gate computing an xor-type (and-type, respectively) operation. A k-*gate* (k^+-*gate*) is a gate of outdegree exactly k (at least k, respectively).

For a circuit C, by $s(C)$ we denote the *size* of C, that is, the number of internal gates of C. By $i(C)$ and $i_1(C)$ we denote the total number of input gates of C and the number of input 1-gates, respectively. For a function $f \in B_n$, by $C_\Omega(f)$ we denote the minimal size of a circuit over Ω computing f.

For two Boolean functions $f, g \in B_n$, the correlation between them is defined as

$$\mathrm{Cor}(f, g) = \left| \Pr_{x \leftarrow \{0,1\}^n}[f(x) = g(x)] - \Pr_{x \leftarrow \{0,1\}^n}[f(x) \neq g(x)] \right| = 2 \left| \frac{1}{2} - \Pr_{x \leftarrow \{0,1\}^n}[f(x) \neq g(x)] \right| .$$

---

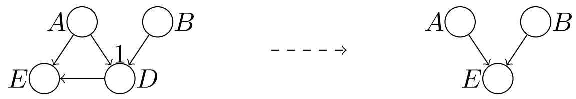
[1] Usually in the literature a source corresponds to a distribution over a subset of $\{0, 1\}^n$. In this paper, we focus only on uniform distributions, so we associate a source with its support.

[2] In this paper, we consider only dispersers and extractors with one bit outputs.

For a function $f \in B_n$, and $0 \leq \varepsilon \leq 1$, by $C_\Omega(f, \varepsilon)$ we denote the minimal size of a circuit over Ω computing function g such that $\mathrm{Cor}(f, g) \geq \varepsilon$.

2.4 Circuit normalization

A gate is called *useless* if it is a 1-gate and is fed by a predecessor of its successor:



In this case E actually computes a binary operation of A and B and this operation can be computed in the gate E directly. This might require to change an operation at E (if this circuit is over U_2 then E still computes an and-type operation of A and B as an xor-type binary function requires three gates in U_2).

By *normalizing* a circuit we mean removing all gates that compute trivial or degenerate operations and removing all useless gates. Note that normalization does not change the function computed by a circuit. It might however change the operations at some gates and outdegrees of some gates (in particular, input gates).

In the proofs of the paper we implicitly assume that if two gates are fed by the same variable then either there is no wire between them or each of the gates feed also some other gate (otherwise, one of the gates would be useless) and hence we do not care about this wire between the gates.

2.5 Circuit complexity measures

A function μ mapping circuits to non-negative real values is called a *circuit complexity measure* if for any circuit C,

- normalization of C does not increase its measure, and
- if $\mu(C) = 0$ then C has no gates.

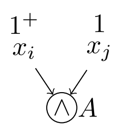
For a fixed circuit complexity measure μ, and function $f \in B_n$, we define $\mu(f)$ to be the minimum value of $\mu(C)$ over circuits C computing f. Similarly, we define $\mu(f, \varepsilon)$ to be the minimum value of $\mu(C)$ over circuits C computing g such that $\mathrm{Cor}(f, g) \geq \varepsilon$.

In this paper, we focus on the following two circuit complexity measures:

- $\mu(C) = s(C) + \alpha \cdot i(C)$ where $\alpha \geq 0$ is a constant;
- $\mu(C) = s(C) + \alpha \cdot i(C) - \sigma \cdot i_1(C)$ where $\alpha \geq 0, \sigma \leq 1$ are constants.

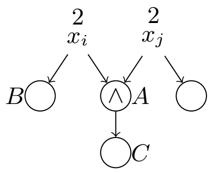
It is not difficult to see that these two functions are indeed circuit complexity measures if $\alpha \geq 0$ and $\sigma \leq 1$. The condition $\sigma \leq 1$ is needed to guarantee that if by removing a degenerate gate we increase the outdegree of a variable, the measure does not increase (an example is given on the next page).

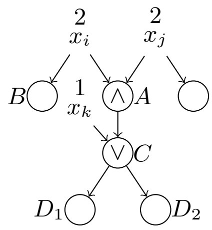
Intuitively we include the term $i(C)$ into the measure to handle cases like the one below (throughout the paper, we use labels above the gates to indicate their outdegree):



In this case, by assigning $x_i \leftarrow 0$ we make the circuit independent of x_j, so the measure is reduced by at least 2α. Usually, our goal is to show that we can find a substitution to a variable that eliminates at least some constant number k of gates, that is, to show a complexity decrease of at least $k + \alpha$. Thus, by choosing a large enough value of α we can always guarantee that $2\alpha \geq \alpha + k$. Thus, in the case above we do not even need to count the number of gates eliminated under the substitution.

The measure $\mu(C) = s(C) + \alpha \cdot i(C) - \sigma \cdot i_1(C)$ allows us to get an advantage of new 1-variables that are introduced during splitting.





For example, by assigning $x_i \leftarrow 0$ in a situation like the one in the left picture we reduce the measure by at least $3 + \alpha + \sigma$. As usual, the advantage comes with a related disadvantage. If, for example, a closer look at the circuit from the left part reveals that it actually looks like as shown on the right, then by assigning $x_i \leftarrow 0$ we introduce a new 1-variable x_j, but also loose one 1-variable (namely, x_k is now a 2-variable). Hence, in this case μ is reduced only by $(3 + \alpha)$ rather than $(3 + \alpha + \sigma)$. That is, our initial estimate was too optimistic. For this reason, when use a measure with $i_1(C)$ we check carefully for each eliminated gate if this elimination increases the degree of a 1-variable.

2.6 Splitting numbers and splitting vectors

Let μ be a circuit complexity measure and C be a circuit. Consider a recursive algorithm solving #SAT on C by repeated substitutions. Assume that at the current step the algorithm chooses k variables $x_1, \ldots, x_k$ and k functions $f_1, \ldots, f_k$ to substitute these variables and branches into 2^k possible situations: $x_1 \leftarrow f_1 \oplus c_1, \ldots, x_k \leftarrow f_k \oplus c_k$ for all possible $c_1, \ldots, c_k \in \{0, 1\}$ (in other words, it partitions the Boolean hypercube $\{0, 1\}^n$ into 2^k subsets).[3] For each substitution, we normalize the resulting circuit. Let us call the 2^k circuits $C_1, \ldots, C_{2^k}$. We say that the current step has a *splitting vector* $v = (a_1, \ldots, a_{2^k})$ w.r.t. μ if for all $i \in [2^k]$, $\mu(C) - \mu(C_i) \geq a_i > 0$. That is, the splitting vector gives a lower bound on the complexity decrease under the considered substitution. The *splitting number* $\tau(v)$ is the unique positive root of the equation $\sum_{i \in [2^k]} x^{-a_i} = 1$.

Splitting vectors and numbers are heavily used to estimate the running time of recursive algorithms. Below we assume that k is bounded by a constant. In all the proofs of this paper either $k = 1$ or $k = 2$, that is, we always estimate the effect of assigning either one or two variables. If an algorithm always splits with a splitting number at most β then its running time is bounded by $O^*(\beta^{\mu(C)})$.[4] To show this one notes that the recursion tree of this algorithm is 2^k-ary and $k = O(1)$ so it suffices to estimate the number of leaves. The

[3] Sometimes it is easier to consider vectors of length that is not a power of 2 too. For example, we can have a branching into three cases: one with one substituted variable, and two with two substituted variables. All the results from this paper can be naturally generalized to this case. For simplicity, we state the results for splitting vectors of length 2^k only.

[4] O^* suppresses factors polynomial in the input length.

number of leaves $T(\mu)$ satisfies the recurrence $T(\mu) \le \sum_{i\in[2^k]} T(\mu - a_i)$ which implies that $T(\mu) = O(\tau(v)^{\mu})$ (we assume also that $T(\mu) = O(1)$ when $\mu = O(1)$). See, e.g., [37] for a formal proof.

For a splitting vector $v = (a_1, \ldots, a_{2^k})$ we define the following related quantities:

$$\overline{v}_{\max} = \max_{i\in[2^k]} \left\{\frac{a_i}{k}\right\}, \quad \overline{v}_{\min} = \min_{i\in[2^k]} \left\{\frac{a_i}{k}\right\}, \quad \overline{v}_{\text{avg}} = \frac{\sum_{i\in[2^k]} a_i}{k2^k}.$$

Intuitively, $\overline{v}_{\max}$ ($\overline{v}_{\min}$, $\overline{v}_{\text{avg}}$) is a (lower bound for) the maximum (minimum, average, respectively) complexity decrease per single substitution.

We will need the following estimates for the splitting numbers. It is known that a balanced binary splitting vector is better than an unbalanced one: $2^{1/a} = \tau(a,a) < \tau(a+b, a-b)$ for $0 < b < a$ (see, e.g., [37]). There is a known upper bound on $\tau(a,b)$.

▶ **Lemma 1.** $\tau(a, b) \le 2^{1/\sqrt{ab}}$.

In the following lemma we provide an asymptotic estimate of their difference.

▶ **Lemma 2** (Gap between $\tau(a_1+b, a_2+b)$ and $\tau((a_1+a_2)/2+b, (a_1+a_2)/2+b) \le 2^{\frac{1}{(a_1+a_2)/2+b}}$). *Let $a_1 > a_2 > 0$, $a' = (a_1 + a_2)/2$ and $\delta(b) = \tau(a_1 + b, a_2 + b) - 2^{\frac{1}{a'+b}}$. Then, $\delta(b) = O((a_1 - a_2)^2/b^3)$ as $b \to \infty$.*

2.7 Azuma's inequality

Following the approach from [13], we use a variant of Azuma's inequality with one-sided boundedness condition in order to obtain average case lower bounds. The standard version of Azuma's inequality requires the difference between two consecutive variables to be bounded, [13] considers the case when the difference takes on only two values but is bounded only from one side. For our results, we need a slightly more general variant of the inequality: the difference between two consecutive variables takes on up to k values and is bounded from one side.

A sequence $X_0, \ldots, X_m$ of random variables is a *supermartingale* if for every $0 \le i < m$, $\mathrm{E}[X_{i+1}|X_i, \ldots, X_0] \le X_i$.

▶ **Lemma 3.** *Let $X_0, \ldots, X_m$ be a supermartingale, let $Y_i = X_i - X_{i-1}$. If $Y_i \le c$ and for fixed values of $(X_0, \ldots, X_{i-1})$, the random variable Y_i is distributed uniformly over at most $k \ge 2$ (not necessarily distinct) values, then for every $\lambda \ge 0$:*

$$\Pr[X_m - X_0 \ge \lambda] \le \exp\left(\frac{-\lambda^2}{2mc^2(k-1)^2}\right).$$

Note that we have an extra factor of $(k-1)^2$ comparing with the normal form of Azuma's inequality, but we do not assume that $X_i - X_{i-1}$ is bounded from below.

3 Toolkit

3.1 Main theorem

In this subsection we prove the main technical theorem that allows us to get circuit complexity lower bounds and #SAT upper bounds.

▶ **Definition 4.** Let $\{v_1, \ldots, v_m\}$ be splitting vectors, and each v_i is a splitting vector of length $2^{t_i} \geq 2$. For a class of circuits Ω (e.g., $\Omega = B_2$ or $\Omega = U_2$), a set of substitutions $\mathcal{S}$, and a circuit complexity measure μ, we write

$$\text{Splitting}(\Omega, \mathcal{S}, \mu) \preceq \{v_1, \ldots, v_m\}$$

as a shortcut for the following statement: For any normalized circuit C from the class Ω one can find in time poly($|C|$) either a substitution[5] from $\mathcal{S}$ whose splitting vector with respect to μ belongs to the set $\{v_1, \ldots, v_m\}$ or a substitution that trivializes the output gate of C. A substitution always trivializes at least one gate (in particular, when we assign a constant to a variable we trivialize an input gate) and eliminates at least one variable.

▶ **Theorem 5.** *If Splitting*$(\Omega, \mathcal{S}, \mu) \preceq \{v_1, \ldots, v_m\}$ *and the longest splitting vector has length* 2^k*, then*

1. *There exists an algorithm solving #SAT for circuits over* Ω *in time* $O^*(\gamma^{\mu(C)})$*, where*

$$\gamma = \max_{i \in [m]} \{\tau(v_i)\} \,.$$

2. *If* $f \in B_n$ *is an* $(\mathcal{S}, n, r)$*-disperser, then*

$$\mu(f) \geq \beta_w \cdot (r - k + 1) \,, \text{ where } \beta_w = \min_{i \in [m]} \{\overline{v_i}_{\max}\} \,.$$

3. *If* $f \in B_n$ *is an* $(\mathcal{S}, n, r, \varepsilon)$*-extractor, then for every* $\mu < \beta_a \cdot r$*,*

$$\mu\left(f, \delta\right) \geq \mu, \text{ where } \beta_a = \min_{i \in [m]} \left\{\overline{v_i}_{\text{avg}}\right\} \text{ and } \beta_m = \min_{i \in [m]} \{\overline{v_i}_{\min}\} \,,$$

$$\delta = \varepsilon + \exp\left(\frac{-(r \cdot \beta_a - \mu)^2}{2r(\beta_a - \beta_m)^2(2^k - 1)^2}\right) \,.$$

3.2 Discussion

Many known lower bounds for circuits with unrestricted depth can be proved using this framework, in particular, the strongest known lower bounds over B_2 and U_2. Schnorr [53] proved a $2n - \Theta(1)$ on C_{B_2} for a wide class of functions using $\mu(C) = s(C)$ and bit fixing substitutions. Stockmeyer [57] proved a $2.5n - \Theta(1)$ lower bound for symmetric functions using $\mu(C) = s(C)$ and a special case of projections: $\{x_i \leftarrow c, \{x_i \leftarrow f, x_j \leftarrow f \oplus 1\}\}$ (the latter "double" substitution essentially fixes $x_i \oplus x_j$ to 1; by applying such a substitution to, say, the majority function one gets the majority function of fewer inputs). Kojevnikov and Kulikov [34] improved the bound by Schnorr to $7n/3 - \Theta(1)$ using the measure $\mu(C) = 3x(C) + 2a(C)$ assigning different weights to xor-gates and and-gates. Demenkov and Kulikov [20] proved a $3n - o(n)$ lower bound for an affine disperser for sublinear dimension using $\mu(C) = s(C) + i(C)$ and affine substitutions. Recently, Find et al. [25] extended this approach to get a $(3 + 1/86)n$ lower bound for the same function using a few additional tricks (while the measure and the set of allowed substitutions are not easy to describe).

For the basis U_2, Schnorr [53] proved that the circuit size of parity is $3n - 3$ using bit fixing substitutions. Zwick [66] proved a $4n - \Theta(1)$ lower bound for symmetric functions using

[5] Here we assume that the circuit obtained from C by the substitution and normalization belongs to Ω too.

bit-fixing substitutions and $\mu(C) = s(C) - i_1(C)$. His measure was then used by Lachish and Raz [38] and by Iwama and Morizumi [32] to prove $4.5n - o(n)$ and $5n - o(n)$ lower bounds for strongly two-dependent functions. Recently, Demenkov et al. [21] gave a simpler proof of a $5n - o(n)$ lower bound for a linear function with $o(n)$ outputs. All these proofs use bit fixing substitutions only, however the case analysis can be simplified using also projections and a measure of the form $\mu = s + \alpha \cdot i$.

At the same time, there are known lower bound proofs that use additional tricks. E.g., Blum [8] to prove a $3n - o(n)$ lower bound over B_2 first considers a few cases when it is easy to remove three gates, and for all the remaining circuits shows a lower bound directly by counting the number of gates using some particular properties of the function under consideration.

Also, upper bounds for SAT and #SAT for various circuits classes (and for many other NP-hard problems) are proved by making substitutions recursively and using a carefully chosen measure to estimate the complexity decrease after substitutions.

The whole framework is a formalization of the following simple idea. To prove a lower bound ζn on circuit size one usually shows that there always exists a substitution $x_i \leftarrow f$ eliminating at least ζ gates from the circuit. By analysing also the complexity decrease under the substitution $x_i \leftarrow f \oplus 1$ one gets an upper bound for #SAT and an average case lower bound. Below we show an easy consequence of this: if one gets a very strong lower bound via short splitting vectors in this framework, then the corresponding #SAT-algorithm is also quite fast. That is, a superlinear circuit lower bound that uses only short splitting vectors in the framework implies a subexponential time (with respect to the size) algorithm for #SAT, which contradicts the Exponential Time Hypothesis.

▶ **Theorem 6.** *If for some set of substitutions $\mathcal{S}$, Splitting$(\Omega, \mathcal{S}, s + \alpha i) \preceq \{(a_1, b_1), \ldots, (a_m, b_m)\}$, such that $\beta_w = \min_{i \in [m]} \max\{a_i, b_i\} = \omega(1)$ then #SAT can be solved in time $O^*(2^{o(s)})$.*

Note that due to the Sparsification Lemma [31] such an algorithm even over the basis U_2 contradicts the Exponential Time Hypothesis.

Although our "positive" results from Theorem 5 hold for splitting vectors of any length, this "negative" result from Theorem 6 holds only for splitting vectors of length 2. The authors do not know how to generalize this result to longer splitting vectors, and leave it as an open question.

4 Bounds for the basis U_2

4.1 Bit fixing substitutions: substituting variables by constants

We start with a well-known case analysis of a $3n - 3$ lower bound for the parity function over U_2 due to Schnorr [53]. Using this case analysis we reprove the bounds given recently by Chen and Kabanets [13] in our framework. The analysis is basically the same though the measure is slightly different. We provide these results mostly as a simple illustration of using the framework.

▶ **Lemma 7.** *Splitting*$(U_2, \{x_i \leftarrow c\}, s + \alpha i) \preceq \{(\alpha, 2\alpha), (3 + \alpha, 3 + \alpha), (2 + \alpha, 4 + \alpha)\}$.

▶ **Corollary 8.** *1. For any $\epsilon > 0$ there exists $\delta = \delta(\epsilon) > 0$ such that #SAT for circuits over U_2 of size at most $(3-\epsilon)n$ can be solved in time $(2-\delta)^n$.*

2. $C_{U_2}(x_1 \oplus \cdots \oplus x_n \oplus c) \geq 3n - 6$.

3. $C_{U_2}\left(x_1 \oplus \cdots \oplus x_n \oplus c, \exp\left(\frac{-(t-9)^2}{18(n-1)}\right)\right) \geq 3n - t$. *This, in particular, implies that* $\mathrm{Cor}(x_1 \oplus \cdots \oplus x_n \oplus c, C)$ *is negligible for any circuit C of size $3n - \omega(\sqrt{n}\log n)$.*

4.2 Projections: substituting variables by constants and other variables

In this subsection, we prove new bounds for the basis U_2.

▶ **Lemma 9.** *For $0 \leq \sigma \leq 1/2$,*

$$\textit{Splitting}(U_2, \{x_i \leftarrow c, x_i \leftarrow x_j \oplus c\}, s + \alpha i - \sigma i_1) \preceq \{(\alpha, 2\alpha), (2\alpha, 2\alpha, 2\alpha, 3\alpha), (3+\alpha+\sigma, 3+\alpha), (4+\alpha+\sigma, 2+\alpha)\}.$$

▶ **Corollary 10.** *1. For any $\epsilon > 0$ there exists $\delta = \delta(\epsilon) > 0$ such that #SAT for circuits over U_2 of size at most $(3.25-\epsilon)n$ can be solved in time $(2-\delta)^n$.*

2. Let $f \in B_n$ be an $\left(n, r(n) = n - \log^{O(1)}(n)\right)$-projections disperser from [40]. Then $C_{U_2}(f) \geq 3.5n - \log^{O(1)}(n)$.

3. Let $f \in B_n$ be an $\left(n, r(n) = n - \sqrt{n}, \varepsilon(n) = 2^{-n^{\Omega(1)}}\right)$-projections extractor from [48]. Then $C_{U_2}(f, \delta) \geq 3.25n - t$, where $\delta = 2^{-n^{\Omega(1)}} + \exp\left(\frac{-(t-10.25\sqrt{n})^2}{190.125(n-\sqrt{n})}\right)$. This, in particular, implies that $\mathrm{Cor}(f, C)$ is negligible for any circuit C of size $3.25n - \omega(\sqrt{n}\log n)$.

5 Bounds for the basis B_2

5.1 Affine substitutions: substituting variables by linear sums of other variables

Here, we again start by reproving the bounds for B_2 by Chen and Kabanets [13] by using the case analysis by Demenkov and Kulikov [20].

▶ **Lemma 11.** *Splitting*$(B_2, \{x_i \leftarrow \oplus_{j\in J} x_j \oplus c\}, \mu = s + \alpha i) \preceq \{(\alpha, 2\alpha), (2+\alpha, 3+\alpha)\}$.

▶ **Corollary 12.** *1. For any $\epsilon > 0$ there exists $\delta = \delta(\epsilon) > 0$ such that #SAT for circuits over B_2 of size at most $(2.5-\epsilon)n$ can be solved in time $(2-\delta)^n$.*

2. Let $f \in B_n$ be an $\left(n, r(n) = n - \log^{O(1)}(n)\right)$-affine disperser from [40]. Then $C_{B_2}(f) \geq 3n - \log^{O(1)}(n)$.

3. Let $f \in B_n$ be an $\left(n, r(n) = n - O(n/\log\log n), \varepsilon(n) = 2^{-n^{\Omega(1)}}\right)$-affine extractor from [39]. Then $C_{B_2}(f, \delta) \geq 2.5n - t$, where $\delta = 2^{-n^{\Omega(1)}} + \exp\left(\frac{-(t-O(n/\log\log n))^2}{O(n)}\right)$. This, in particular, implies that $\mathrm{Cor}(f, C)$ is negligible for any circuit C of size $2.5n - \omega(n/\log\log n)$.

5.2 Quadratic substitutions: substituting variables by degree 2 polynomials of other variables

▶ **Lemma 13.** *For* $0 \le \sigma \le 1/5$,

$$\textit{Splitting}(B_2, \{x_i \leftarrow p\colon \deg(p) \le 2\}, s + \alpha i - \sigma i_1) \preceq \{(\alpha, 2\alpha), (2\alpha, 2\alpha, 2\alpha, 3\alpha), (3 + \alpha - 2\sigma, 3 + \alpha - 2\sigma), (3 + \alpha + \sigma, 2 + \alpha)\}\,.$$

▶ **Corollary 14.** **1.** *For any* $\epsilon > 0$ *there exists* $\delta = \delta(\epsilon) > 0$ *such that #SAT for circuits over* B_2 *of size at most* $(2.6 - \epsilon)n$ *can be solved in time* $(2 - \delta)^n$.

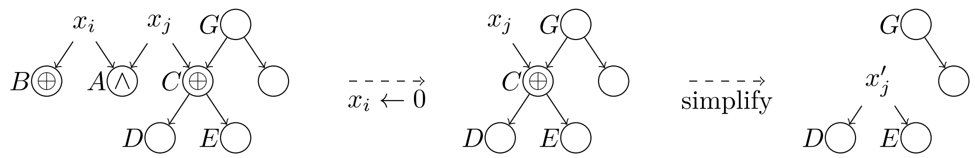
2. *Let* $f \in B_n$ *be an* $(n, r(n) = n - o(n))$*-quadratic disperser. Then* $C_{B_2}(f) \ge 3n - o(n)$.

3. *Let* $f \in B_n$ *be an* $\left(n, r(n) = n - o(n), \varepsilon(n) = 2^{-\omega(\log n)}\right)$*-quadratic extractor. Then* $C_{B_2}(f, \delta) \ge 2.6n - t$, *where* $\delta = 2^{-n^{\Omega(1)}} + \exp\left(\frac{-(t-7.8(n-r(n)))^2}{121.68r(n)}\right)$. *This, in particular, implies that* $\mathrm{Cor}(f, C)$ *is negligible for any circuit* C *of size* $2.6n - g(n)$ *for some* $g(n) = o(n)$.

▶ Remark 1. Note that it is an open problem to find an explicit construction of quadratic disperser or extractor over $\mathbb{F}_2$ with $r = n - o(n)$. Any disperser for a slightly more general definition of quadratic varieties would also imply a new worst case lower bound [26].

▶ Remark 2. Note that the upper bound for #SAT can be improved using the following "forbidden trick", that is, a simplification rule that reduces the size of a circuit without changing the number of its satisfying assignments, but changes the function computed by the circuit.

In the proof of Lemma 13 set $\sigma = 0$ (that is, do not account for 1-variables). The set of splitting vectors then turn into By inspecting all the cases, we see that the splitting vector $(3 + \alpha, 2 + \alpha)$ only appears in one case. We can handle this case differently: split on x_i. When A is trivialized, x_j becomes a 1-variable feeding an xor-gate. It is not difficult to show that by replacing this gate with a new variable x'_j one gets a circuit with the same number of satisfying assignments.



This additional trick gives us the following set of splitting vectors: $\{(\alpha, 2\alpha), (2\alpha, 2\alpha, 2\alpha, 3\alpha), (3+\alpha, 3+\alpha), (4+\alpha, 2+\alpha)\}$. These splitting numbers give an algorithm solving #SAT in $(2-\delta(\epsilon))^n$ for B_2-circuits of size at most $(3 - \epsilon)n$ for $\epsilon > 0$.

Note that such a simplification rule does not fit into our framework since it *changes the function computed by a circuit.* It would be interesting to adjust the framework to allow such kind of simplifications (probably, by incorporating some new parameter to the measure).

6 Open problems

There are three natural questions left open in this paper.

1. Prove that a superlinear circuit lower bound in this framework violates the Exponential Time Hypothesis.

2. Give an explicit construction of quadratic dispersers (see Remark 1).
3. Adjust the framework to allow using natural simplification rules like replacing an xor gate fed by a 1-variable for both upper bounds for #SAT and lower bounds for circuit size (see Remark 2).

Acknowledgement. We would like to thank the anonymous reviewers for their helpful comments.

References

1. Kazuyuki Amano and Atsushi Saito. A nonuniform circuit class with multilayer of threshold gates having super quasi polynomial size lower bounds against NEXP. In *Proceedings of the 9th International Conference on Language and Automata Theory and Applications (LATA)*, pages 461–472, 2015.
2. Kazuyuki Amano and Atsushi Saito. A satisfiability algorithm for some class of dense depth two threshold circuits. *IEICE Transactions*, 98-D(1):108–118, 2015.
3. Kazuyuki Amano and Jun Tarui. A well-mixed function with circuit complexity $5n$: Tightness of the Lachish-Raz-type bounds. *Theor. Comput. Sci.*, 412(18):1646–1651, 2011.
4. Paul Beame, Russell Impagliazzo, and Srikanth Srinivasan. Approximating AC^0 by small height decision trees and a deterministic algorithm for $\#\mathrm{AC}^0$ SAT. In *Proceedings of the 27th Conference on Computational Complexity (CCC)*, pages 117–125, 2012.
5. Eli Ben-Sasson and Ariel Gabizon. Extractors for polynomial sources over fields of constant order and small characteristic. *Theory of Computing*, 9:665–683, 2013.
6. Eli Ben-Sasson and Swastik Kopparty. Affine dispersers from subspace polynomials. *SIAM J. Comput.*, 41(4):880–914, 2012.
7. Eli Ben-Sasson and Emanuele Viola. Short PCPs with projection queries. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP), Part I*, pages 163–173, 2014.
8. Norbert Blum. A Boolean function requiring $3n$ network size. *Theor. Comput. Sci.*, 28:337–345, 1984.
9. Harry Buhrman, Lance Fortnow, and Thomas Thierauf. Nonrelativizing separations. In *Proceedings of the 13th Annual IEEE Conference on Computational Complexity (CCC)*, pages 8–12, 1998.
10. Venkatesan T. Chakaravarthy and Sambuddha Roy. Oblivious symmetric alternation. In *Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 230–241, 2006.
11. Venkatesan T. Chakaravarthy and Sambuddha Roy. Arthur and Merlin as oracles. *Computational Complexity*, 20(3):505–558, 2011.
12. Ruiwen Chen. Satisfiability algorithms and lower bounds for Boolean formulas over finite bases. In *Proceedings of the 40th International Symposium on Mathematical Foundations of Computer Science (MFCS), Part II*, pages 223–234, 2015.
13. Ruiwen Chen and Valentine Kabanets. Correlation bounds and #SAT algorithms for small linear-size circuits. In *Proceedings of the 21st International Conference on Computing and Combinatorics (COCOON)*, pages 211–222, 2015.
14. Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining circuit lower bound proofs for meta-algorithms. *Computational Complexity*, 24(2):333–392, 2015.

15 Ruiwen Chen, Valentine Kabanets, and Nitin Saurabh. An improved deterministic #SAT algorithm for small De Morgan formulas. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS), Part II*, pages 165–176, 2014.

16 Ruiwen Chen and Rahul Santhanam. Improved algorithms for sparse MAX-SAT and MAX-k-CSP. In *Proceedings of the 18th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 33–45, 2015.

17 Ruiwen Chen, Rahul Santhanam, and Srikanth Srinivasan. Average-case lower bounds and satisfiability algorithms for small threshold circuits. In *Proceedings of the 31th Conference on Computational Complexity (CCC)*, pages 1:1–1:35, 2016.

18 Benny Chor, Oded Goldreich, Johan Håstad, Joel Friedman, Steven Rudich, and Roman Smolensky. The bit extraction problem of t-resilient functions (preliminary version). In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 396–407, 1985.

19 Gil Cohen and Igor Shinkar. The complexity of DNF of parities. In *Proceedings of the 7th Innovations in Theoretical Computer Science (ITCS) Conference*, pages 47–58, 2016.

20 Evgeny Demenkov and Alexander S. Kulikov. An elementary proof of a $3n - o(n)$ lower bound on the circuit complexity of affine dispersers. In *Proceedings of the 36th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 256–265, 2011.

21 Evgeny Demenkov, Alexander S. Kulikov, Olga Melanich, and Ivan Mihajlin. New lower bounds on circuit size of multi-output functions. *Theory of Computing Systems*, 56(4):630–642, 2015.

22 Yevgeniy Dodis. *Exposure-resilient cryptography.* PhD thesis, Massachusetts Institute of Technology, 2000.

23 Zeev Dvir. Extractors for varieties. *Computational Complexity*, 21(4):515–572, 2012.

24 Zeev Dvir, Ariel Gabizon, and Avi Wigderson. Extractors and rank extractors for polynomial sources. *Computational Complexity*, 18(1):1–58, 2009.

25 Magnus Gausdal Find, Alexander Golovnev, Edward Hirsch, and Alexander Kulikov. A better-than-$3n$ lower bound for the circuit complexity of an explicit function. *Electronic Colloquium on Computational Complexity (ECCC)*, TR15-166, 2015.

26 Alexander Golovnev and Alexander S. Kulikov. Weighted gate elimination: Boolean dispersers for quadratic varieties imply improved circuit lower bounds. In *Proceedings of the 7th Innovations in Theoretical Computer Science (ITCS) Conference*, pages 405–411, 2016.

27 Alexander Golovnev, Alexander S. Kulikov, Alexander Smal, and Suguru Tamaki. Circuit size lower bounds and #sat upper bounds through a general framework. *Electronic Colloquium on Computational Complexity (ECCC)*, TR16-022, 2016.

28 Johan Håstad. On the correlation of parity and small-depth circuits. *SIAM J. Comput.*, 43(5):1699–1708, 2014.

29 Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC^0. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 961–972, 2012.

30 Russell Impagliazzo, Ramamohan Paturi, and Stefan Schneider. A satisfiability algorithm for sparse depth two threshold circuits. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 479–488, 2013.

31 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

32 Kazuo Iwama and Hiroki Morizumi. An explicit lower bound of $5n - o(n)$ for Boolean circuits. In *Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 353–364, 2002.

33 Hamid Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP), Part I*, pages 749–760, 2015.

34 Arist Kojevnikov and Alexander S. Kulikov. Circuit complexity and multiplicative complexity of Boolean functions. In *Proceedings of the 6th Conference on Computability in Europe (CiE)*, pages 239–245, 2010.

35 Ilan Komargodski and Ran Raz. Average-case lower bounds for formula size. In *Proceedings of the 45th Symposium on Theory of Computing (STOC)*, pages 171–180, 2013.

36 Ilan Komargodski, Ran Raz, and Avishay Tal. Improved average-case lower bounds for De Morgan formula size. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 588–597, 2013.

37 Oliver Kullmann. Fundaments of branching heuristics. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 205–244. IOS Press, 2009.

38 Oded Lachish and Ran Raz. Explicit lower bound of $4.5n - o(n)$ for Boolean circuits. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 399–408, 2001.

39 Xin Li. A new approach to affine extractors and dispersers. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity (CCC)*, pages 137–147, 2011.

40 Xin Li. Improved two-source extractors, and affine extractors for polylogarithmic entropy. *Electronic Colloquium on Computational Complexity (ECCC)*, TR15-125, 2015.

41 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

42 Dániel Marx. Consequences of ETH: Tight bounds for various problems. In *Fine-Grained Complexity and Algorithm Design Boot Camp*, 2015. “https://simons.berkeley.edu/talks/daniel-marx-2015-09-03” (abstract, slides and archived video).

43 Atsuki Nagao, Kazuhisa Seto, and Junichi Teruyama. A moderately exponential time algorithm for k-IBDD satisfiability. In *Proceedings of the 14th International Symposium, on Algorithms and Data Structures (WADS)*, pages 554–565, 2015.

44 Sergey Nurk. An $O(2^{0.4058m})$ upper bound for circuit SAT. Technical report, PDMI, 2009.

45 Igor Carboni Oliveira. Algorithms versus circuit lower bounds. *Electronic Colloquium on Computational Complexity (ECCC)*, TR13-117, 2013.

46 Ramamohan Paturi, Michael E. Saks, and Francis Zane. Exponential lower bounds for depth three boolean circuits. *Computational Complexity*, 9(1):1–15, 2000.

47 Wolfgang J. Paul. A $2.5n$-lower bound on the combinational complexity of Boolean functions. *SIAM J. Comput.*, 6(3):427–443, 1977.

48 Anup Rao. Extractors for low-weight affine sources. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity (CCC)*, pages 95–101, 2009.

49 Takayuki Sakai, Kazuhisa Seto, Suguru Tamaki, and Junichi Teruyama. Bounded depth circuits with weighted symmetric gates: Satisfiability, lower bounds and compression. In *Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2016, to appear.

50 Rahul Santhanam. Circuit lower bounds for Merlin-Arthur classes. *SIAM J. Comput.*, 39(3):1038–1061, 2009.

51 Rahul Santhanam. Fighting perebor: New and improved algorithms for formula and QBF satisfiability. In *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 183–192, 2010.

52 Rahul Santhanam. Ironic complicity: Satisfiability algorithms and circuit lower bounds. *Bulletin of the EATCS*, 106:31–52, 2012.

53 Claus-Peter Schnorr. Zwei lineare untere schranken für die komplexität boolescher funktionen. *Computing*, 13(2):155–171, 1974.

54 Kazuhisa Seto and Suguru Tamaki. A satisfiability algorithm and average-case hardness for formulas over the full binary basis. *Computational Complexity*, 22(2):245–274, 2013.

55 Ronen Shaltiel. Dispersers for affine sources with sub-polynomial entropy. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 247–256, 2011.

56 Claude E. Shannon. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal*, 28(1):59–98, 1949.

57 Larry J. Stockmeyer. On the combinational complexity of certain symmetric Boolean functions. *Mathematical Systems Theory*, 10:323–336, 1977.

58 Avishay Tal. #SAT algorithms from shrinkage. *Electronic Colloquium on Computational Complexity (ECCC)*, TR15-114, 2015.

59 Fengming Wang. NEXP does not have non-uniform quasipolynomial-size ACC circuits of $o(\log\log n)$ depth. In *Proceedings of the 8th Annual Conference on Theory and Applications of Models of Computation (TAMC)*, pages 164–170, 2011.

60 Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013.

61 Ryan Williams. Natural proofs versus derandomization. In *Proceedings of the 45th ACM Symposium on Theory of Computing Conference (STOC)*, pages 21–30, 2013.

62 Ryan Williams. Algorithms for circuits and circuits for algorithms. In *Proceedings of the 29th Annual IEEE Conference on Computational Complexity (CCC)*, pages 248–261, 2014.

63 Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. In *Proceedings of the 46th Symposium on Theory of Computing (STOC)*, pages 194–202, 2014.

64 Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2, 2014.

65 Amir Yehudayoff. Affine extractors over prime fields. *Combinatorica*, 31(2):245–256, 2011.

66 Uri Zwick. A $4n$ lower bound on the combinational complexity of certain symmetric Boolean functions over the basis of unate dyadic Boolean functions. *SIAM J. Comput.*, 20(3):499–505, 1991.