Electronic Colloquium on Computational Complexity, Report No. 22 (2016)

Circuit size lower bounds and #SAT upper bounds through a general framework

Alexander Golovnev* Alexander S. Kulikov† Alexander V. Smal‡
Suguru Tamaki§

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 refine the approach by Chen and Kabanets and provide a general framework for proving worst/average case lower bounds for circuits and upper bounds for #SAT. 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 technical 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.

*New York University alexgolovnev@gmail.com. Research is partially supported by NSF grant 1319051.

†St. Petersburg Department of Steklov Institute of Mathematics of the Russian Academy of Sciences kulikov@logic.pdmi.ras.ru. Research is partially supported by the Government of the Russian Federation (grant 14.Z50.31.0030).

‡St. Petersburg Department of Steklov Institute of Mathematics of the Russian Academy of Sciences smal@logic.pdmi.ras.ru. Research is partially supported by the Government of the Russian Federation (grant 14.Z50.31.0030).

§Kyoto University tamak@kuis.kyoto-u.ac.jp. Supported in part by MEXT KAKENHI (24106003); JSPS KAKENHI (25240002, 26330011); the John Mung Advanced Program of Kyoto University. Part of the work performed while the author was at Department of Computer Science and Engineering, University of California, San Diego and the Simons Institute for the Theory of Computing, Berkeley.

ISSN 1433-8092

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 $\mathrm{P} \neq \mathrm{NP}$. However, despite of many efforts [53, 47, 57, 9, 20, 26, 66, 31, 4], 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 parity and its complement [37, 31].

Going to larger complexity classes, it is known that the classes MA/1 [50], O_2^p [11], and $\mathrm{P}^{\mathrm{prMA}}$ [12] require circuits of superlinear size and the class MAEXP [10] 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 $\mathrm{NE} \not\subseteq \mathcal{C}$, depending on $\mathcal{C}$ and the running time of the algorithm. The approach has been strengthened and simplified by subsequent work [59, 61, 63, 8, 32], see also excellent surveys [52, 45, 62] on this topic.

Williams' result inspired lots of work on satisfiability algorithms for various circuit classes [29, 63, 16, 3, 2, 43, 17, 49, 58]. In addition to satisfiability algorithms, several papers [51, 28, 5, 54, 15, 13] also obtained average-case lower bounds (also known as correlation bounds, see [34, 35, 27]) by investigating the analysis of algorithms instead of just applying Williams' result that yields worst-case lower bounds. In particular, Chen and Kabanets [14] 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.

1.2 Our Techniques and Results

The main qualitative contribution of the paper is a refinement of the approach by Chen and Kabanets [14]. We show that many known (worst case and average case) circuit lower bounds and #SAT upper bounds are immediate consequences of the underlying case analysis (the case analysis is intended to show that for any circuit one can find a substitution that reduces the complexity of a given circuit by a sufficient amount). In particular, the bounds in [14] are obtained by using the case analysis of [53] and [20]. We make this transition explicit by establishing a framework that allows to derive lower bounds for circuits and upper bounds for #SAT as few lines corollaries from the corresponding case analysis. We use new circuit complexity measures and substitution types to improve known bounds for circuits size and #SAT in this framework.

The main quantitative contribution of the paper is the following new bounds (improving the bounds from [14]):

- 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;
- faster than 2^n #SAT-algorithms for circuits over U_2 and B_2 of size at most $3.24n$ and $2.99n$, respectively.

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 [30] 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., [40, 41].

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 curcuit lower bounds [19, 20, 25]. There are several known constructions of dispersers [7, 55] and extractors [65, 38, 6, 39] that are resistant to $n - o(n)$ substitutions.
4. The class of quadratic substitutions generates a special case of *polynomial sources* [24, 6] 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.

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.

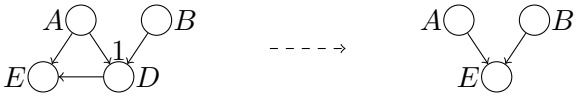
For two Boolean functions $f, g \in B_n$, the correlation between them is defined as

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

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 predecessor:



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

[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.

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 gate 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.

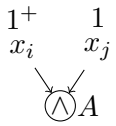
For a fixed circuit complexity measure μ, and function $f \in B_n$, we define $\mu(f)$ to be the minimum value of $\mu(C)$ over circuits C computing f. Similarly, we define $\mu(f, \varepsilon)$ to be the minimum value of $\mu(C)$ over circuits C computing g such that $\mathrm{Cor}(f, g) \geq \varepsilon$.

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.

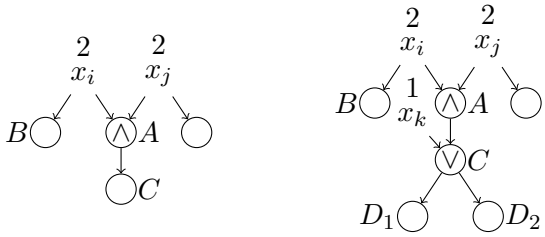
It is not difficult to see that these two functions are indeed circuit complexity measures if $\alpha \geq 0$ and $\sigma \leq 1$. The condition $\sigma \leq 1$ is needed to guarantee that if by removing a degenerate gate we increase the outdegree of a variable, the measure does not increase (an example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

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 gate 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 number of leaves $T(\mu)$ satisfies the recurrence $T(\mu) \leq \sum_{i \in [2^k]} T(\mu - a_i)$ which implies that $T(\mu) = O(\tau(v)^\mu)$ (we assume also that $T(\mu) = O(1)$ when $\mu = O(1)$). See, e.g., [36] for a formal proof.

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

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

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

[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.

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., [36]). There is a known upper bound on $\tau(a,b)$.

Lemma 1 (Lemma 5.8 in [36])**.** $\tau(a,b) \le 2^{1/\sqrt{ab}}$.

In the following lemma we provide an asymptotic estimate of their difference (the proof is given in Appendix on p. 25).

Lemma 2 (Gap between $\tau(a_1+b, a_2+b)$ and $\tau((a_1+a_2)/2+b, (a_1+a_2)/2+b) = 2^{\frac{1}{(a_1+a_2)/2+b}}$)**.** *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 [14], we use a variant of Azuma's inequality with one-sided boundness 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, [14] 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. We provide a proof of this inequality, which is just an adjustment of proofs from [42, 1, 14] (the proof is given in Appendix on page 25).

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 (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 3.1. *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*

$$\mathit{Splitting}(\Omega, \mathcal{S}, \mu) \preceq \{v_1, \ldots, v_m\}^5$$

[5] Note that each v_i is a splitting vector of length $2^{t_i} \ge 2$.

as a shortcut for the following statement: For any normalized circuit C from the class Ω one can find in time $\mathrm{poly}(|C|)$ *either a substitution*[6] *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 4. *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*

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

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

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

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

Proof. We present a proof for a special case when all splitting vectors have length 2 (i.e., $k = 1$): $\{v_1, \ldots, v_m\} = \{(a_1, b_1), \ldots, (a_m, b_m)\}$. This makes the exposition simpler, and it is easy to see that the general statement follows by the same argument. In this case,

$$\beta_w = \min_{i \in [m]} \{\max\{a_i, b_i\}\}, \beta_a = \min_{i \in [m]} \left\{\frac{a_i + b_i}{2}\right\}, \beta_m = \min_{i \in [m]} \{\min\{a_i, b_i\}\}.$$

1. Consider the following branching algorithm for #SAT. We describe the algorithm as a branching tree, each node of which contains a Boolean circuit and a set of currently made substitutions. The root of the tree is $(C, \emptyset)$ — the input circuit and an empty set of substitutions. The nodes where the circuit is trivialized are called leaves. At each internal node (a node that is not a leaf) the algorithm finds in polynomial time substitutions $x_i \leftarrow f$ and $x_i \leftarrow f \oplus 1$ guaranted by the theorem statement. Then the algorithm recursively calls itself on two circuits obtained from the current one by substituting $x_i \leftarrow f$ and $x_i \leftarrow f \oplus 1$. That is, the algorithm adds to the current node (C, S) two children $(C|x_i \leftarrow f, S \cup \{x_i \leftarrow f\})$ and $(C|x_i \leftarrow f \oplus 1, S \cup \{x_i \leftarrow f \oplus 1\})$. Note that the statement guarantees that the substitutions $x_i \leftarrow f$ and $x_i \leftarrow f \oplus 1$ either give us an (a_i, b_i)-splitting for some i (i.e., decrease the measure μ by at least a_i in one branch, and b_i in the other one), or trivialize the circuit and produce two leaves.

   At each leaf the algorithm counts the number V of satisfying assignments: If the formula is constant zero, then $V = 0$, otherwise, $V = 2^v$, where v is the number of variables in the current

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

formula. Indeed, for each assignment to the v variables, there exists a unique assignment to the rest of the variables (via the substitutions at the leaf), and the circuit remains constant 1. Since substitutions $x_i \leftarrow f$ and $x_i \leftarrow f \oplus 1$ lead to different assignments to the input variables, the leaves of the branching tree correspond to disjoint sets of assignments. Therefore, the total number of satisfying assignments of the original circuit is the sum of satisfying assignments at the leaves of the tree. Since the running time of the algorithm at each node is polynomial, the total running time is bounded from above by $O^*\left(\gamma^{\mu(C)}\right)$, where $\gamma = \max_{i \in [m]}\{\tau(a_i, b_i)\}$.

2. For every pair of integers (n, r) such that $n \geq r \geq 0$, let $\mathcal{F}_{n,r} \subseteq B_n$ denote the class of functions from $\{0,1\}^n$ to $\{0,1\}$ that are not constant after any r substitutions from $\mathcal{S}$. We show that for every $f \in \mathcal{F}_{n,r}$, $\mu(f) \geq \beta_w \cdot (r-k+1)$. This claim implies the theorem statement, since we start with a function of n variables that is not constant after $r(n)$ substitutions.

   The proof of the claim proceeds by induction on r. For $r < k$ the statement is trivial. Now assume that $r \geq k$. Consider substitutions $x_i \leftarrow f$ and $x_i \leftarrow f \oplus 1$ guaranteed by the lemma statement. Now select a value of $c \in \{0,1\}$ in such a way that the substitution $x_i \leftarrow f \oplus c$ reduces the measure by at least β_w. Consider the function g of $n-1$ variables which is f restricted to $x_i \leftarrow f$. By the theorem statement, $\mu(f) \geq \beta_w + \mu(g)$, and by the induction hypothesis, $\mu(g) \geq \beta_w \cdot (r-1)$. Therefore, $\mu(f) \geq \beta_w \cdot r$.

3. Let us consider a circuit C such that $\mu(C) \leq \beta_a \cdot r$. Consider the branching tree from the 1st part of the proof. At each node of the branching tree let us uniformly at random choose a child we proceed to. Let δ_i be the random variable that equals to the measure decrease at ith step (ith level of the branching tree, where 0 corresponds to the root). For $i \geq 0$, define the random variable

$$X_i = (i+1) \cdot \beta_a - \sum_{j=0}^{i} \delta_j.$$

   Let us show that the variable X_i is a supermartingale:

$$\mathrm{E}[X_i | X_{i-1}, \ldots, X_0] = i \cdot \beta_a - \sum_{j=0}^{i-1} \delta_j + (\beta_a - \mathrm{E}[\delta_i | X_{i-1}, \ldots, X_0]) = X_{i-1} + (\beta_a - \mathrm{E}[\delta_i]) \leq X_{i-1}.$$

   Let $Y_i = X_i - X_{i-1}$. Then Y_i is distributed uniformly over at most 2^k values, and $Y_i \leq \beta_a - \beta_m$. Now let $\lambda = \beta_a \cdot r - \mu(C)$. Then, by Lemma 3:

$$\Pr[X_r - X_0 \geq \lambda] \leq \exp\left(\frac{-\lambda^2}{2r(\beta_a - \beta_m)^2(2^k-1)^2}\right).$$

   Now we want to bound from above the correlation between f and the function given by the branching tree. Note that all leaves of the tree that have depth smaller than r altogether give correlation at most ε with the extractor f (since each of these leaves defines an $(\mathcal{S}, n, r)$ source). Now let us count the number of leaves at the depth at least r. There are at most 2^r possible leaves, but each of them survives till the rth level only with probability $\Pr[X_r - X_0 \geq \lambda]$. Indeed, if $X_r - X_0 < \lambda$, then $\sum_{j=1}^{r} \delta_j > \mu(C)$, which means that the function becomes constant before the rth level. Therefore, there are at most $2^r \cdot \Pr[X_r - X_0 \geq \lambda]$ leaves

at the depth at least r. Since each leaf at the depth r has r inputs fixed, it covers at most 2^{n-r} points of the Boolean cube. Therefore, the total correlation is bounded from above by:

$$\mathrm{Cor}(f,C) \le \varepsilon + \exp\left(\frac{-\lambda^2}{2r(\beta_a-\beta_m)^2(2^k-1)^2}\right) = \varepsilon + \exp\left(\frac{-(r\cdot\beta_a-\mu(C))^2}{2r(\beta_a-\beta_m)^2(2^k-1)^2}\right).$$

□

3.2 Discussion

Many known lower bounds for circuits with unrestricted depth can be proved using this framework, in particular, 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 the sum of x_i+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 [33] 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 (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 bit-fixing substitutions and $\mu(C)=s(C)-i_1(C)$. His measure was then used by Lachish and Raz [37] and by Iwama and Morizumi [31] 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 [9] 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 cn on circuit size one usually shows that there always exists a substitution $x_i \leftarrow f$ eliminating at least c gates from the circuit. By analyzing 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 5. *If for some set of substitutions $\mathcal{S}$,*

$$\mathit{Splitting}(\Omega, \mathcal{S}, s + \alpha i) \preceq \{(a_1, b_1), \ldots, (a_m, b_m)\},$$

such that $\beta_w = \min_{i \in [m]} \max\{a_i, b_i\} = \omega(1)$ then #SAT can be solved in time $O^*(2^{o(s)})$.

Proof. Since we consider substitutions that trivialize at least one gate and eliminate at least one variable, we may assume w.l.o.g. that $a_i \geq 1 + \alpha$ and $b_i \geq \omega(1)$ for all $i \in [m]$. Then, any splitting (guaranteed by the theorem statement) w.r.t. to $s(C)$ is at most $\tau(1, \omega(1))$. From Lemma 1, $\tau(1, x) \leq 2^{\frac{1}{\sqrt{x}}}$. We conclude that the corresponding algorithm for #SAT always splits with a splitting number at most $2^{\frac{1}{\omega(1)}}$ and hence has running time $2^{o(s)}$. □

Note that due to the Sparsification Lemma [30] such an algorithm even over the basis U_2 contradicts the Exponential Time Hypothesis.

Although our "positive" results from Theorem 4 hold for splitting vectors of any length, this "negative" result from Theorem 5 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

In this and the following sections, we prove known and new circuit lower bounds and upper bounds for #SAT. As discussed already, the main ingredient of all proofs is the case analysis showing the existence of a substitution reducing the measure by a sufficient amount. Usually, in such proofs we argue as follows: take a gate A and make a substitution trivializing A; this eliminates A and all its successors. However it might be the case that A is the output gate and so does not have any successors. This means that we are at the end of the gate elimination process or at the leaf of a recursion tree. This, in turn, means that we do not need to estimate the current measure decrease. For this reason, in all the proofs below we assume implicitly that if we trivialize a gate then it is not the output gate.

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 [14] 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. The proof of the following case analysis is given in Appendix on page 26.

Lemma 6. $\mathit{Splitting}(U_2, \{x_i \leftarrow c\}, s + \alpha i) \preceq \{(\alpha, 2\alpha), (3 + \alpha, 3 + \alpha), (2 + \alpha, 4 + \alpha)\}$.

Corollary 7. *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 $3n - \omega(\sqrt{n \log n})$.

Proof. 1. First note that for large enough α, we have $\tau(\alpha, 2\alpha) < \tau(3+\alpha, 3+\alpha) = 2^{\frac{1}{3+\alpha}} < \tau(2+\alpha, 4+\alpha)$. Let $\gamma(\alpha) = \tau(2+\alpha, 4+\alpha) - 2^{\frac{1}{3+\alpha}}$. By Lemma 2, $\gamma(\alpha) = O(1/\alpha^3)$ holds. The running time of the algorithm is at most

$$\begin{aligned}(\tau(2+\alpha, 4+\alpha))^{s+\alpha n} &\leq \left(2^{\frac{1}{3+\alpha}}(1+\gamma(\alpha))\right)^{s+\alpha n} \leq 2^{\frac{s+\alpha n}{3+\alpha}} 2^{(s+\alpha n)\gamma(\alpha)\log_2 e} \\ &\leq 2^{\frac{(3-\epsilon)n+\alpha n}{3+\alpha} + O(n/\alpha^2)} \leq (2-\delta)^n\end{aligned}$$

for some $\delta > 0$ if we set $\alpha = c/\epsilon$ for large enough $c > 0$.

2. The parity function does not become constant under any $n-1$ substitutions of variables to constants. Lemma 6 guarantees that for $\alpha = 3$ we can always assign a constant to a variable so that $s + 3i$ is reduced by at least 6. Hence for any circuit C over U_2 computing parity, $s(C) + 3n \geq 6(n-1)$ implying $s(C) \geq 3n - 6$. (In fact, it can be shown by a slightly more careful analysis that $C_{U_2}(x_1 \oplus \cdots \oplus x_n) = 3n - 3$.)

3. Let us consider a circuit C of size at most $3n - t$, that is, $\mu(C) \leq (3n - t) + \alpha n$. Now we fix $\alpha = 6$, then $\beta_a = \min\{9, 9, 9\} = 9, \beta_m = \min\{6, 9, 8\} = 6$.

   We use the third item of Theorem 4 with $k = 1$, $r = n - 1$, $\varepsilon = 0$, $\mu = (3n - t + 6n)$, which gives us

$$\delta = \exp\left(\frac{-(9(n-1) - (3n - t + 6n))^2}{18(n-1)}\right) = \exp\left(\frac{-(t-9)^2}{18(n-1)}\right).$$

□

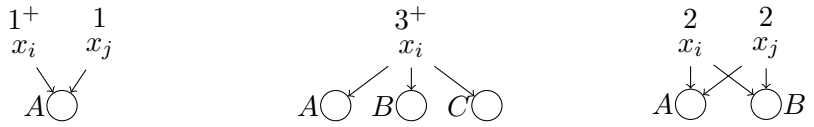
4.2 Projections: substituting variables by constants and other variables

In this subsection, we prove new bounds for the basis U_2. The two main ideas leading to improved bounds are using projections to handle the Case 3 below and using 1-variables to get better estimates for complexity decrease (this trick was used by Zwick [66] and then by [37, 31]).

Lemma 8. *For $0 \leq \sigma \leq 1/2$,*

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

Proof. Note that for every eliminated gate we decrease the measure by at least $1 - \sigma \geq 1/2$, if some gate becomes constant we decrease the measure by at least 1.



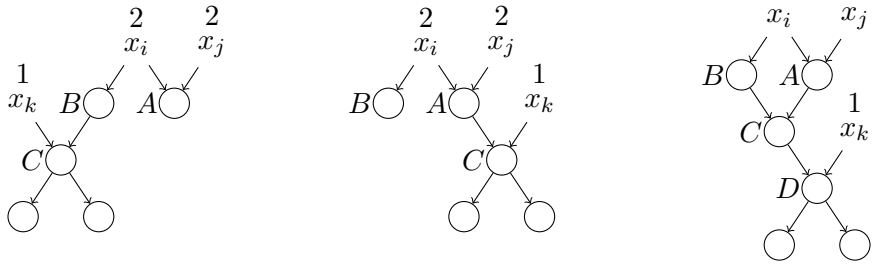
Case 1 Case 2 Case 3

Case 1. There is a top gate A fed by 1-variable x_j. Assigning x_i a constant we trivialize the gate A in one of the branches and loose the dependence on x_j. Thus, we get at least $(\alpha, 2\alpha)$ splitting.

Case 2. There is a variable x_i of degree at least 3. Neither of A, B, C is fed by a 1-variable otherwise we would be in the Case 1. When we assign x_i a constant, gates A, B, and C become constant in one of the branches. Hence in one of the branches we eliminate also at least one extra gate. Thus, we get at least $(4+\alpha-\sigma, 3+\alpha)$ splitting vector which dominates $(3+\alpha+\sigma, 3+\alpha)$ (since $\sigma \leq 1/2$).

Case 3. There are two 2-variables that feed the same two top gates. Let the gates A and B compute Boolean functions $f_A(x_i, x_j) = (x_i \oplus a_A)(x_j \oplus b_A) \oplus c_A$ and $f_B(x_i, x_j) = (x_i \oplus a_B)(x_j \oplus b_B) \oplus c_B$ respectively. If $a_A = a_B$ or $b_A = b_B$ then we assign $x_i \leftarrow a_A$ or $x_j \leftarrow b_A$ respectively and make both gates constant. Otherwise, $f_B(x_i, x_j) = (x_i \oplus a_A \oplus 1)(x_j \oplus b_A \oplus 1) \oplus c_B$. It is easy to see that if $x_i \oplus a_A \oplus x_j \oplus b_A = 1$ then both functions are constant. Hence, the substitution $x_i \leftarrow a_A \oplus x_j \oplus b_A \oplus 1$ makes A and B constant as well. In both cases there is a substitution that makes A and B constant and therefore eliminates the dependence on x_j, so we get at least $(\alpha, 2\alpha)$ splitting vector.

Case 4. There are three gates that are fed by two 2-variables.



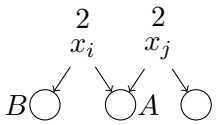
Case 4.1 Case 4.2 Case 4.3

Case 4.1. Gate B is a 1-gate with a successor C that is a 2^+-gate fed by the 1-variable x_k. If we assign constants to x_j and x_k we eliminate the dependence on x_j as well in one of the branches, so we get $(2\alpha, 2\alpha, 2\alpha, 3\alpha)$ splitting.

Case 4.2. Gate A is a 1-gate with a successor C that is a 2^+-gate fed by 1-variable x_k. Analogous to the previous case if we assign constants to x_i and x_j we eliminate the dependence on x_k in one of the branches, so we get again $(2\alpha, 2\alpha, 2\alpha, 3\alpha)$ splitting.

Case 4.3. Gates A and B and its common successor C are 1-gates and the only successor of C is a 2-gate fed by 1-variable x_k. When we assign constants to x_k gate D becomes constant in one of the branches, hence, gates A, B, C become unnecessary and we loose the dependence on variable x_i. So we have at least $(\alpha, 2\alpha)$ splitting.

Case 4.4. None of three previous cases apply.



Previous cases ruled out the possibility that A or B has the only successor that contributes only $1-\sigma$ to the measure decrease: we know that each of A and B is either a 2^+-gate and its successors contribute at least $2(1-\sigma) \geq 1$ or a 1-gate with a successor which is not a 2^+-gate fed by a 1-variable and it contributes at least 1. We also know, that when A and B are 1-gates with a common successor, this successor is not a 2^+-gate fed by a 1-variable, and hence it contributes at least 1.

Therefore, if we assign x_i a constant each of A and B becomes constant in one of the branches, so the successors of A and B either contribute at least 1 in both branches or contribute at least 2 in one of the branches. In addition, x_j becomes a 1-variable in the branch where A trivializes. Thus, we get either $(3+\alpha+\sigma, 3+\alpha)$ or $(4+\alpha+\sigma, 2+\alpha)$.

□

Corollary 9. *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 [39]. 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 $3.25n - \omega(\sqrt{n \log n})$.

Proof. 1. Let $\sigma = 1/2$. First note that for large enough α, we have

$$\tau(\alpha, 2\alpha) < \tau(2\alpha, 2\alpha, 2\alpha, 3\alpha) < \tau(3.25+\alpha, 3.25+\alpha) = 2^{\frac{1}{3.25+\alpha}} < \tau(3.5+\alpha, 3+\alpha) < \tau(4.5+\alpha, 2+\alpha).$$

Let $\gamma(\alpha) = \tau(4.5+\alpha, 2+\alpha) - 2^{\frac{1}{3.25+\alpha}}$. By Lemma 2, $\gamma(\alpha) = O(1/\alpha^3)$ holds. The running time of the algorithm is at most

$$\begin{aligned}(\tau(4.5+\alpha, 2+\alpha))^{s+\alpha n} &\leq \left(2^{\frac{1}{3.25+\alpha}}(1+\gamma(\alpha))\right)^{s+\alpha n} \leq 2^{\frac{s+\alpha n}{3.25+\alpha}} 2^{(s+\alpha n)\gamma(\alpha)\log_2 e} \\ &\leq 2^{\frac{(3.25-\epsilon)n+\alpha n}{3.25+\alpha} + O(n/\alpha^2)} \leq (2-\delta)^n\end{aligned}$$

for some $\delta > 0$ if we set $\alpha = c/\epsilon$ for large enough $c > 0$.

2. Lemma 8 guarantees that for $\alpha = 7$, $\sigma = 0.5$ one can always make an affine substitution reducing $s + 7i$ by at least 10.5. The function f is resistant to $r(n)$ such substitutions. Hence for a circuit C computing f, $s(C) + 7n \geq 10.5 r(n)$.

3. Let us consider a circuit C of size at most $3.25n-t$, that is, $\mu(C) \le (3.25n-t)+\alpha n$. Now we fix $\alpha = 7$, $\sigma = 0.5$, then $\beta_a = \min\{10.5, 15.75, 10.25, 10.25\} = 10.25, \beta_m = \min\{7, 7, 10, 9\} = 7$.

   We use the third item of Theorem 4 with $k = 2$, $r = n-\sqrt{n}$, $\varepsilon = 2^{-n^{\Omega(1)}}$, $\mu = (3.25n-t+7n)$, which gives us

$$\delta = 2^{-n^{\Omega(1)}} + \exp\left(\frac{-(10.25(n-\sqrt{n}) - (10.25n-t))^2}{2(n-\sqrt{n})\cdot 3.25^2 \cdot 3^2}\right) = 2^{-n^{\Omega(1)}} + \exp\left(\frac{-(t-10.25\sqrt{n})^2}{190.125(n-\sqrt{n})}\right).$$

□

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 [14] by using the case analysis by Demenkov and Kulikov [20]. The proof of the following lemma is given in Appendix on page 26.

Lemma 10. *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 11. *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 [39]. Then $C_{B_2}(f) \ge 3n - \log^{O(1)}(n)$.

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 [38]. Then $C_{B_2}(f, \delta) \ge 2.5n - t$, 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 $2.5n - \omega(n/\log\log n)$.

Proof. 1. First note that for large enough α, we have

$$\tau(\alpha, 2\alpha) < \tau(2.5+\alpha, 2.5+\alpha) = 2^{\frac{1}{2.5+\alpha}} < \tau(2+\alpha, 3+\alpha).$$

Let $\gamma(\alpha) = \tau(2+\alpha, 3+\alpha) - 2^{\frac{1}{2.5+\alpha}}$. By Lemma 2, $\gamma(\alpha) = O(1/\alpha^3)$ holds. The running time of the algorithm is at most

$$\begin{aligned}(\tau(2+\alpha, 3+\alpha))^{s+\alpha n} &\le \left(2^{\frac{1}{2.5+\alpha}}(1+\gamma(\alpha))\right)^{s+\alpha n} \le 2^{\frac{s+\alpha n}{2.5+\alpha}} 2^{(s+\alpha n)\gamma(\alpha)\log_2 e} \\ &\le 2^{\frac{(2.5-\epsilon)n+\alpha n}{2.5+\alpha} + O(n/\alpha^2)} \le (2-\delta)^n\end{aligned}$$

for some $\delta > 0$ if we set $\alpha = c/\epsilon$ for large enough $c > 0$.

2. Lemma 10 guarantees that for $\alpha = 3$ one can always make an affine substitution reducing $s+3i$ by at least 6. The function f is resistant to $r(n)$ such substitutions. Hence for a circuit C computing f, $s(C) + 3n \ge 6r(n)$.

3. Let us consider a circuit C of size at most $2.5n - t$, that is, $\mu(C) \le (2.5n - t) + \alpha n$. Now we fix $\alpha = 5$, then $\beta_a = \min\{7.5, 7.5\} = 7.5, \beta_m = \min\{5, 7\} = 5$.

   We use the third item of Theorem 4 with $k = 1$, $r = r(n)$, $\varepsilon = \varepsilon(n)$, $\mu = (2.5n - t + 5n)$, which gives us

$$\delta = \varepsilon(n) + \exp\left(\frac{-(7.5r(n) - (7.5n - t))^2}{12.5r(n)}\right) = \varepsilon(n) + \exp\left(\frac{-(t - 7.5(n - r(n)))^2}{12.5r(n)}\right).$$

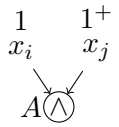
□

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

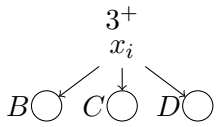
Lemma 12. *For $0 \le \sigma \le 1/5$,*

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

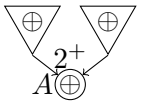
Proof. Fix any topological order of a given circuit C and let A be the first gate in this ordering which is not a 1-xor (if there is no such gate then all the gates in C are 1-xors hence C computes an affine function and we can trivialize it with a single affine substitution). Then each input of A is a tree of xors, that is, a subcircuit consisting of 1-xors only. When we do an affine substitution to some variable that feed an xor-tree, we rebuild the tree and reduce the number of gates in it by at least one (it is explained in details in the proof of Lemma 10).



Case 1



Case 2



Case 3

(In all the pictures of this proof we show only the type of the gates but not the actual functions computed at them.)

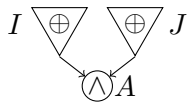
Case 1. A is a top and-gate fed by a 1-variable x_i. Similarly to the Case 1 of Lemma 8 we get $(\alpha, 2\alpha)$.

Case 2. There is a variable x_i of degree at least 3. Neither of B, C, D is an and-gate fed by a 1-variable otherwise we would be in the Case 1. If, say, B is an xor 2^+-gate fed by the 1-variable x_k we can trivialize it by an affine substitution $x_i \leftarrow x_k \oplus c$ and eliminate two variables in both branches, so we get $(2\alpha, 2\alpha)$. Otherwise we assign x_i a constant and eliminate three gates in every branch, all the gates contribute 1 to the measure decrease. Thus, we get at least $(3 + \alpha, 3 + \alpha)$ which dominates $(3 + \alpha - 2\sigma, 3 + \alpha - 2\sigma)$.

Case 3. A is 2^+-xor. Let A compute $c_A \oplus \bigoplus_{i \in I} x_i$ and $I \subseteq \{1, \ldots, n\}$, $|I| \ge 2$. If all x_i, $i \in I$, are 1-variables then for any $j \in I$ a substitution $x_j \leftarrow c \oplus \bigoplus_{i \in I \setminus \{j\}} x_i$ eliminates the dependence on at least one 1-variable, so we get $(2\alpha, 2\alpha)$. Otherwise, there is at least one 2-variable x_i,

$i \in I$. Substituting $x_i \leftarrow c \oplus \bigoplus_{k \in I \setminus \{i\}} x_k$ we eliminate three gates in both branches, so we get at least $(3 + \alpha - 2\sigma, 3 + \alpha - 2\sigma)$.

Case 4. A is an and-gate which is not a top gate.

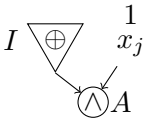


Let $I, J \subset \{1, \ldots, n\}$ be the sets of indices of variables in the left and in the right xor-trees respectively. W.l.o.g., we assume that $|I| > 1$, i.e. there is at least one gate in the left xor-tree feeding A.

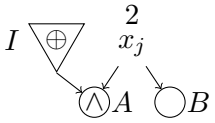
Case 4.1. There is a 1-variable x_i in the left tree. An affine substitution $x_j \leftarrow c \oplus \bigoplus_{k \in J \setminus \{j\}} x_k$ for some $j \in J$ eliminates two variables in one of the branches: the variable x_i becomes unnecessary in the branch where A becomes constant. The splitting is at least $(\alpha, 2\alpha)$.

Case 4.2. There is at least one gate in the right tree, i.e. $|J| \geq 2$. We apply an affine substitution $x_i \leftarrow c \oplus \bigoplus_{k \in I \setminus \{i\}} x_k$ for some $i \in I$ and eliminate four gates in one branch and two in the other one. This gives $(4 + \alpha - \sigma, 2 + \alpha)$ which dominates $(3 + \alpha + \sigma, 2 + \alpha)$.

Case 4.3. The right tree consists of only one variable x_j.



Case 4.3.1



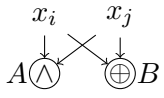
Case 4.3.2

Case 4.3.1. x_j is a 1-variable. An affine substitution $x_i \leftarrow c \oplus \bigoplus_{k \in I \setminus \{i\}} x_k$ for some $i \in I$ eliminates x_j in the branch where A becomes constant, so we get at least $(\alpha, 2\alpha)$.

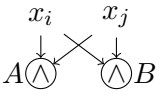
Case 4.3.2. x_j is a 2-variable. Assigning x_j a constant we eliminate four gates in one branch and two in the other one. Gate B do not introduce new an 1-variable: if B is a 2^+-gate fed by 1-variable we would be either in the Case 1 or in the Case 3. The splitting on x_j gives at least $(4 + \alpha - \sigma, 2 + \alpha)$ which dominates $(3 + \alpha + \sigma, 2 + \alpha)$.

Case 5. A is a top and-gate fed by 2-variables x_i and x_j.

Case 5.1. Variables x_i and x_j feed the same two gates.



Case 5.1.1



Case 5.1.2

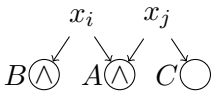
Case 5.1.1. B is an xor-gate. An affine substitution $x_i \leftarrow x_j \oplus c$ eliminates the dependence on x_j in the branch where A becomes constant, so we get at least $(\alpha, 2\alpha)$.

Case 5.1.2. B is an and-gate. Similarly to the Case 3 of the proof of Lemma 8 we get $(\alpha, 2\alpha)$.

Case 5.2. Variables x_i and x_j feed three gates: A, B, and C.

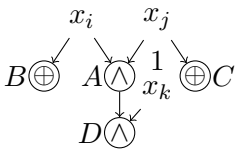
Note that in the following cases eliminating gates B and C we can not kill a 1-variable, otherwise we would be either in the Case 1 of in the Case 3.

Case 5.2.1. x_i and x_j feed two and-gates. W.l.o.g., B is an and-gate.

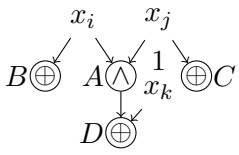


Assigning a constant to x_i we trivialize gates A and B in one of the branches, so we eliminate either four gates in one branch and two in the other one, or three gates in both branches. This gives either $(4+\alpha-2\sigma, 2+\alpha)$ or $(3+\alpha-\sigma, 3+\alpha-\sigma)$, which dominate $(3+\alpha+\sigma, 2+\alpha)$ and $(3+\alpha-2\sigma, 3+\alpha-2\sigma)$ respectively.

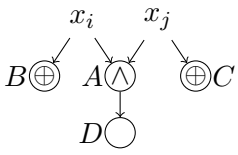
Case 5.2.2. Both B and C are xor-gates.



Case 5.2.2.1



Case 5.2.2.2



Case 5.2.2.3

Case 5.2.2.1. A is a 1-gate and its only successor D is an and-gate fed by the 1-variable x_k. Assigning constants to x_i and x_j we eliminate also the dependence on x_k in one of the branches. We get at least $(2\alpha, 2\alpha, 2\alpha, 3\alpha)$.

Case 5.2.2.2. A is a 1-gate and its only successor D is an xor-gate fed by the 1-variable x_k. Let gate A computes function $(x_i \oplus a_A)(x_j \oplus b_A) \oplus c_A$. An affine substitution $x_k \leftarrow (x_i \oplus a_A)(x_j \oplus b_A) \oplus c$ makes D constant and eliminates at least three gates in each branch; in addition x_i and x_j become 1-variables. So, we get at least $(3+\alpha+\sigma, 3+\alpha+\sigma)$ which dominates $(3+\alpha+\sigma, 2+\alpha)$. Note that x_k only feeds gate D which is now constant so we do not need to replace x_k by a subcircuit computing $(x_i \oplus a_A)(x_j \oplus b_A) \oplus c$.

Case 5.2.2.3. A is either a 2^+-gate or a 1-gate with the only successor D which is not fed by a 1-variable. Assigning x_i a constant we eliminate three gates in one branch and two in the other one, in addition x_j becomes a 1-variable in the branch where A becomes constant. Gate D do not introduce new 1-variables, otherwise we would be in one of the previous two cases. Thus, we get $(3+\alpha+\sigma, 2+\alpha)$.

□

Corollary 13. *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 $C_{B_2}(f) \geq 3n - o(n)$.

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 $C_{B_2}(f, \delta) \geq 2.6n - t$, 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 $g(n) = o(n)$.

Proof. 1. Let $\sigma = 1/5$. First note that for large enough α, we have

$$\tau(\alpha, 2\alpha) < \tau(2\alpha, 2\alpha, 2\alpha, 3\alpha) < \tau(2.6+\alpha, 2.6+\alpha) = 2^{\frac{1}{2.6+\alpha}} < \tau(3.5+\alpha, 3+\alpha) < \tau(3.2+\alpha, 2+\alpha).$$

Let $\gamma(\alpha) = \tau(3.2+\alpha, 2+\alpha) - 2^{\frac{1}{2.6+\alpha}}$. By Lemma 2, $\gamma(\alpha) = O(1/\alpha^3)$ holds. The running time of the algorithm is at most

$$\begin{aligned}(\tau(3.2+\alpha,2+\alpha))^{s+\alpha n} &\leq \left(2^{\frac{1}{2.6+\alpha}}(1+\gamma(\alpha))\right)^{s+\alpha n} \leq 2^{\frac{s+\alpha n}{2.6+\alpha}} 2^{(s+\alpha n)\gamma(\alpha)\log_2 e} \\ &\leq 2^{\frac{(2.6-\epsilon)n+\alpha n}{2.6+\alpha}+O(n/\alpha^2)} \leq (2-\delta)^n\end{aligned}$$

for some $\delta > 0$ if we set $\alpha = c/\epsilon$ for large enough $c > 0$.

2. Lemma 12 guarantees that for $\alpha = 6$, $\sigma = 0$ one can always make an affine substitution reducing $s + 6i$ by at least 9. The function f is resistant to $r(n)$ such substitutions. Hence for a circuit C computing f, $s(C) + 6n \geq 9r(n)$.

3. Let us consider a circuit C of size at most $2.6n - t$, that is, $\mu(C) \leq (2.6n - t) + \alpha n$. Now we fix $\alpha = 5.2$, $\sigma = 0.2$, then $\beta_a = \min\{7.8, 11.7, 7.8, 7.8\} = 7.8$, $\beta_m = \min\{5.2, 5.2, 7.8, 7.2\} = 5.2$.

   We use the third item of Theorem 4 with $k = 2$, $r = r(n)$, $\varepsilon = \varepsilon(n)$, $\mu = (2.6n - t + 5.2n)$, which gives us

$$\delta = \varepsilon(n) + \exp\left(\frac{-(7.8r(n) - (7.8n - t))^2}{2r(n) \cdot 2.6^2 \cdot 3^2}\right) = \varepsilon(n) + \exp\left(\frac{-(t - 7.8(n - r(n)))^2}{121.68r(n)}\right).$$

□

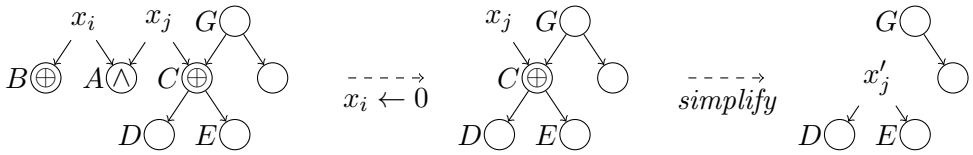
Remark 1. *Note that it is an open problem to find an explicit construction of quadratic disperser or extractor over $\mathbb{F}_2$ with $r = n - o(n)$. Any disperser for a slightly more general definition of quadratic varieties would also imply a new worst case lower bound [26].*

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 12 set $\sigma = 0$ (that is, do not account for 1-variables). The set of splitting vectors then turn into

$$\{(\alpha, 2\alpha), (2\alpha, 2\alpha, 2\alpha, 3\alpha), (3+\alpha, 3+\alpha), (3+\alpha, 2+\alpha)\}.$$

By inspecting all the cases, we see that the splitting vector $(3+\alpha, 2+\alpha)$ only appears in the the Case 5.2.2. 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] N. Alon and J. H. Spencer. *The probabilistic method*. John Wiley & Sons, 2004.

[2] K. Amano and A. 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.

[3] K. Amano and A. Saito. A satisfiability algorithm for some class of dense depth two threshold circuits. *IEICE Transactions*, 98-D(1):108–118, 2015.

[4] K. Amano and J. Tarui. A well-mixed function with circuit complexity $5n$: Tightness of the Lachish-Raz-type bounds. *Theor. Comput. Sci.*, 412(18):1646–1651, 2011.

[5] P. Beame, R. Impagliazzo, and S. 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.

[6] E. Ben-Sasson and A. Gabizon. Extractors for polynomial sources over fields of constant order and small characteristic. *Theory of Computing*, 9:665–683, 2013.

[7] E. Ben-Sasson and S. Kopparty. Affine dispersers from subspace polynomials. *SIAM J. Comput.*, 41(4):880–914, 2012.

[8] E. Ben-Sasson and E. 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.

[9] N. Blum. A Boolean function requiring $3n$ network size. *Theor. Comput. Sci.*, 28:337–345, 1984.

[10] H. Buhrman, L. Fortnow, and T. Thierauf. Nonrelativizing separations. In *Proceedings of the 13th Annual IEEE Conference on Computational Complexity (CCC)*, pages 8–12, 1998.

[11] V. T. Chakaravarthy and S. Roy. Oblivious symmetric alternation. In *Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 230–241, 2006.

[12] V. T. Chakaravarthy and S. Roy. Arthur and Merlin as oracles. *Computational Complexity*, 20(3):505–558, 2011.

[13] R. 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.

[14] R. Chen and V. 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.

[15] R. Chen, V. Kabanets, A. Kolokolova, R. Shaltiel, and D. Zuckerman. Mining circuit lower bound proofs for meta-algorithms. *Computational Complexity*, 24(2):333–392, 2015.

[16] R. Chen, V. Kabanets, and N. 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.

[17] R. Chen and R. 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.

[18] B. Chor, O. Goldreich, J. Håstad, J. Friedman, S. Rudich, and R. 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] G. Cohen and I. Shinkar. The complexity of DNF of parities. In *Proceedings of the 7th Innovations in Theoretical Computer Science (ITCS) Conference*, pages 47–58, 2016.

[20] E. Demenkov and A. 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] E. Demenkov, A. S. Kulikov, O. Melanich, and I. Mihajlin. New lower bounds on circuit size of multi-output functions. *Theory of Computing Systems*, 56(4):630–642, 2015.

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

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

[24] Z. Dvir, A. Gabizon, and A. Wigderson. Extractors and rank extractors for polynomial sources. *Computational Complexity*, 18(1):1–58, 2009.

[25] M. G. Find, A. Golovnev, E. Hirsch, and A. 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] A. Golovnev and A. 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] J. Håstad. On the correlation of parity and small-depth circuits. *SIAM J. Comput.*, 43(5):1699–1708, 2014.

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

[29] R. Impagliazzo, R. Paturi, and S. 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.

[30] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[31] K. Iwama and H. 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.

[32] H. Jahanjou, E. Miles, and E. Viola. Local reductions. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP), Part I*, pages 749–760, 2015.

[33] A. Kojevnikov and A. 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.

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

[35] I. Komargodski, R. Raz, and A. 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.

[36] O. Kullmann and H. Luckhardt. Algorithms for SAT/TAUT decision based on various measures. Preprint, 71 pages, 1998.

[37] O. Lachish and R. 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.

[38] X. 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.

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

[40] D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

[41] D. 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).

[42] B. Maurey. Espaces de Banach: Construction de suites symetriques. *C.R. Acad. Sci. Paris Ser. A-B.*, 288:679–681, 1979.

[43] A. Nagao, K. Seto, and J. 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] S. Nurk. An $O(2^{0.4058m})$ upper bound for circuit SAT. Technical report, PDMI, 2009.

[45] I. C. Oliveira. Algorithms versus circuit lower bounds. *Electronic Colloquium on Computational Complexity (ECCC)*, TR13-117, 2013.

[46] R. Paturi, M. E. Saks, and F. Zane. Exponential lower bounds for depth three boolean circuits. *Computational Complexity*, 9(1):1–15, 2000.

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

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

[49] T. Sakai, K. Seto, S. Tamaki, and J. Teruyama. A satisfiability algorithm for depth-2 circuits with a symmetric gate at the top and AND gates at the bottom. *Electronic Colloquium on Computational Complexity (ECCC)*, TR15-136, 2015.

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

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

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

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

[55] R. 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] C. E. Shannon. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal*, 28(1):59–98, 1949.

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

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

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

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

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

[62] R. 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] R. 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] R. Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2, 2014.

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

[66] U. 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.

Appendix

Here we provide omitted proofs of technical lemmas.

Lemma 2 (Gap between $\tau(a_1+b,a_2+b)$ and $\tau((a_1+a_2)/2+b,(a_1+a_2)/2+b) = 2^{\frac{1}{(a_1+a_2)/2+b}}$)**.** *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 $b \to \infty$.*

Proof. Let $x = \tau(a_1 + b, a_2 + b)$, then by definition we have

$$1 = \frac{1}{x^{a_1+b}} + \frac{1}{x^{a_2+b}} = \frac{x^{-(a_1-a_2)/2} + x^{(a_1-a_2)/2}}{x^{a'+b}}.$$

Since

$$x = 2^{\frac{1}{a'+b}} + \delta(b) = 1 + \frac{\ln 2}{a'+b} + \delta(b) + O\left(\frac{1}{(a'+b)^2}\right)$$

and

$$(1+\epsilon)^{(a_1-a_2)/2} = 1 + (a_1-a_2)\epsilon/2 + (a_1-a_2)(a_1-a_2-1)\epsilon^2/4 + O(\epsilon^3),$$

we have

$$x^{-(a_1-a_2)/2} + x^{(a_1-a_2)/2} = 2 + \frac{(a_1-a_2)^2}{2}\left(\frac{\ln 2}{a'+b} + \delta(b)\right)^2 + O\left(\left(\frac{\ln 2}{a'+b} + \delta(b)\right)^3\right).$$

We also have

$$x^{a'+b} = 2\left(1 + \delta(b)/2^{\frac{1}{a'+b}}\right)^{a'+b} = 2\left(1 + (a'+b)\delta(b)/2^{\frac{1}{a'+b}} + O(\delta(b)^2)\right).$$

By the definition of x, we have

$$\lim_{b\to\infty} \frac{(a_1-a_2)^2 \ln^2 2}{2b^2}/(2b\delta(b)) = 1.$$

This implies

$$\delta(b) = \frac{(a_1-a_2)^2 \ln^2 2}{4b^3} + o(1/b^3).$$

□

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 (not necessarily distinct) values, then for every $\lambda \ge 0$:*

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

Proof. For any $t > 0$,

$$\Pr\left[X_m - X_0 \ge \lambda\right] = \Pr\left[\sum_{i=1}^m Y_i \ge \lambda\right] = \Pr\left[\exp\left(t \cdot \sum_{i=1}^m Y_i\right) \ge e^{\lambda t}\right] \le e^{-\lambda t} \cdot \mathrm{E}\left[\exp\left(t \cdot \sum_{i=1}^m Y_i\right)\right].$$

First we show that for any $t > 0$, $\mathrm{E}[e^{tY_i}] \le \exp\left(t^2c^2(k-1)^2/2\right)$. Since $\{X_i\}$ is a supermartingale, $\mathrm{E}[Y_i|X_{i-1}, \ldots, X_0] \le 0$. W.l.o.g., assume that $\mathrm{E}[Y_i|X_{i-1}, \ldots, X_0] = 0$, otherwise we can increase the values of negative Y_i's which only increases the objective function $\mathrm{E}[e^{tY_i}]$. Note that $\mathrm{E}[Y_i] = 0, Y_i \le c$ and Y being uniform over k values imply that $|Y_i| \le c(k-1)$. Let

$$h(y) = \frac{e^{tc(k-1)} + e^{-tc(k-1)}}{2} + \frac{e^{tc(k-1)} - e^{-tc(k-1)}}{2} \cdot y$$

be the line going through points $(-c(k-1), h(-c(k-1)))$ and $(c(k-1), h(c(k-1)))$. From convexity of e^{tY}, $e^{tY} \leq h(y)$ for $|y| \leq c(k-1)$. Thus,

$$\mathrm{E}[e^{tY_i}] \leq \mathrm{E}[h(Y_i)] = h\left(\mathrm{E}[Y_i]\right) = h(0) = \cosh(tc(k-1)) \leq \exp\left(t^2c^2(k-1)^2/2\right),$$

where the last inequality $\cosh(x) \leq \exp\left(x^2/2\right)$ for $x > 0$ can be proven by comparing the Taylor series of the two functions.

Now,

$$\mathrm{E}\left[\exp\left(t \cdot \sum_{i=1}^{m} Y_i\right)\right] = \mathrm{E}\left[\exp\left(t \cdot \sum_{i=1}^{m-1} Y_i\right) \cdot \mathrm{E}\left[\exp\left(Y_m | X_{m-1}, \ldots, X_0\right)\right]\right] \leq$$
$$\mathrm{E}\left[\exp\left(t \cdot \sum_{i=1}^{m-1} Y_i\right)\right] \cdot \exp\left(t^2c^2(k-1)^2/2\right) \leq \exp\left(mt^2c^2(k-1)^2/2\right),$$

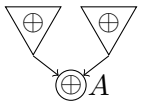
which for $t = \lambda/mc^2(k-1)^2$ implies $\Pr[X_m - X_0 \geq \lambda] \leq \exp\left(\frac{-\lambda^2}{2mc^2(k-1)^2}\right)$. □

Lemma 6. *Splitting*$(U_2, \{x_i \leftarrow c\}, s + \alpha i) \preceq \{(\alpha, 2\alpha), (3+\alpha, 3+\alpha), (2+\alpha, 4+\alpha)\}$.

Proof. Let A be a top-gate computing $(x_i \oplus a)(x_j \oplus b) \oplus c$ where x_i, x_j are input variables and $a, b, c \in \{0, 1\}$ are constants. If $\mathrm{out}(x_i) = \mathrm{out}(x_j) = 1$ we split on x_i. When $x_i \leftarrow a$ the gate A trivializes and the resulting circuit becomes independent on x_j. This gives $(\alpha, 2\alpha)$. Assume now that $\mathrm{out}(x_i) \geq 2$. Denote by B the other successor of x_i and let C, D be successors of A, B, respectively. Note that $B \neq C$ since the circuit is normalized but it might be the case that $C = D$. We then split on x_i. Both A and B trivialize in at least one of the branches and their successors are also eliminated. This gives us either $(3+\alpha, 3+\alpha)$ or $(2+\alpha, 4+\alpha)$. (Note if A and B trivialize in the same branch and $C = D$ then we counted C twice in the analysis above. However in this case C also trivializes so all its successors are also eliminated.) □

Lemma 10. *Splitting*$(B_2, \{x_i \leftarrow \oplus_{j \in J} x_j \oplus c\}, \mu = s + \alpha i) \preceq \{(\alpha, 2\alpha), (2+\alpha, 3+\alpha)\}$.

Proof. Fix any topological ordering of a given circuit C and let A be the first gate in this ordering which is not a 1-xor (if there is no such gate then all the gates in C are 1-xors hence C computes an affine function and we can trivialize it with a single affine substitution). Let P and Q be inputs to A. Each of P and Q is computed by a tree of xors, that is, a subcircuit consisting of 1-xors only. Since each gate in such a tree has outdegree 1, it is not used in any other part of the circuit. Also, both P and Q might as well be input gates.



In any case, we can trivialize, say, P by an affine substitution. If P is an input gate this can be done simply by assigning the corresponding variable a constant. If P is an internal gate then it computes a sum $\bigoplus_{j \in J} x_j \oplus c$. To trivialize it, we select any variable $i \in J$ and make a substitution $x_i \leftarrow \bigoplus_{j \in J \setminus \{i\}} x_j \oplus c'$. This clearly makes P constant. To remove x_i from the circuit we replace

the whole tree for P by a new tree computing $\bigoplus_{j\in J\setminus\{i\}} x_j \oplus c'$ (at this point, we use essentially the fact that all the gates in the tree computing P were needed to compute P only; hence when P is trivialized all these gates may be removed safely). We then replace all occurrences of x_i by this new tree. The new tree has one gate less than the old one. So when P is an internal gate, by trivializing it we eliminate a variable and the gate P itself.

Case 1. A is a 2^+-xor. Then A itself is computed by a tree of 1-xors. Trivializing it gives $(3+\alpha, 3+\alpha)$.

Case 2. A is an and-gate and one of its inputs (say, P) is an internal gate. We trivialize P. In both branches we eliminate A and P, but in one of them A is trivialized so we eliminate also its successors. This gives $(2+\alpha, 3+\alpha)$.

Case 3. A is an and-gate fed by two variables x_i and x_j.

Case 3.1. The outdegree of one of them (say, x_i) is at least 2. Then splitting on x_i gives $(2+\alpha, 3+\alpha)$.

Case 3.2. $\mathrm{out}(x_i) = \mathrm{out}(x_j) = 1$. Then splitting on x_i is $(\alpha, 2\alpha)$.

□

ECCC ISSN 1433-8092

http://eccc.hpi-web.de