Exponential Complexity of Satisfiability Testing for Linear-Size Boolean Formulas

Evgeny Dantsin and Alexander Wolpert

Department of Computer Science, Roosevelt University
430 S. Michigan Av., Chicago, IL 60605, USA
{edantsin,awolpert}@roosevelt.edu

Abstract. The *exponential complexity* of the satisfiability problem for a given class of Boolean circuits is defined to be the infimum of constants α such that the problem can be solved in time $poly(m)\,2^{\alpha n}$, where m is the circuit size and n is the number of input variables [IP01]. We consider satisfiability of linear Boolean formula over the full binary basis and we show that the corresponding exponential complexities are "interwoven" with those of k-CNF SAT in the following sense. For any constant c, let f_c be the exponential complexity of the satisfiability problem for Boolean formulas of size at most cn. Similarly, let s_k be the exponential complexity of k-CNF SAT. We prove that for any c, there exists a k such that $f_c \leq s_k$. Since the Sparsification Lemma [IPZ01] implies that for any k, there exists a c such that $s_k \leq f_c$, we have $\sup_c\{f_c\} = \sup_k\{s_k\}$. (In fact, we prove this equality for a larger class of linear-size circuits that includes Boolean formulas.) Our work is partly motivated by two recent results. The first one is about a similar "interweaving" between linear-size circuits of constant depth and k-CNFs [SS12]. The second one is that satisfiability of linear-size Boolean formulas can be tested exponentially faster than in $O(2^n)$ time [San10, ST12].

1 Introduction

Assuming $\mathbf{P} \neq \mathbf{NP}$, it is still unknown how to classify **NP**-complete problems by their complexity. For example, is it possible to test satisfiability of 3-CNFs in subexponential time? The conjecture known as the Exponential Time Hypothesis (ETH) states that it is not possible [IP01]. Or, is it possible to test satisfiability of Boolean circuits exponentially faster than using the trivial enumeration of all assignments? Questions like these seem far away from being resolved, even though this line of research has produced useful insights, see surveys in [DH09, PP10].

Exponential Complexity. A natural approach to the complexity classification of problems in **NP** is to use the notion of exponential complexity [IP01, CP09]. In this paper, we restrict ourselves to Boolean satisfiability problems. Let $\mathcal{C}$ be a class of circuits and let $\mathcal{C}$ SAT be the satisfiability problem for circuits of $\mathcal{C}$. The *exponential complexity* of $\mathcal{C}$ SAT is the infimum of constants α such that there is an algorithm that solves this problem in time $poly(m)\,2^{\alpha n}$, where

P.G. Spirakis and M. Serna (Eds.): CIAC 2013, LNCS 7878, pp. 110–121, 2013.

m is the circuit size and n is the number of input variables, see Section 2 for details. Many deep and interesting results on exponential complexity and related questions are relevant to this paper, but due to the space limit, we mention here only few of them, namely known results on the exponential complexity for k-CNFs, linear-size circuits of constant depth, and linear-size Boolean formulas. To describe these results, we use the following notation (throughout the paper, n denotes the number of variables in a circuit):

- s_k is the exponential complexity of the satisfiability problem for k-CNFs; $s_\infty = \sup_k\{s_k\}$;
- r_c^d is the exponential complexity of the satisfiability problem for circuits of depth at most d and size at most cn; $r_\infty^d = \sup_c\{r_c^d\}$;
- f_c is the exponential complexity of the satisfiability problem for Boolean formulas of size at most cn over the full binary basis; $f_\infty = \sup_c\{f_c\}$.

In these terms, ETH is the conjecture that $s_3 > 0$. Impagliazzo and Paturi [IP01] proved that if ETH is true then the sequence $\{s_k\}$ increases infinitely often. Using the sparsification technique, it was shown that s_k remains the same if the class of k-CNFs is restricted to k-CNFs of linear size [IPZ01]. Known upper bounds on s_k have the form $1 - c/k$, where c is a constant [DH09], and it is a challenging open question whether $s_\infty = 1$ or $s_\infty < 1$. The Strong Exponential Time Hypothesis (SETH) states that $s_\infty = 1$ [IP01, CIP09].

An upper bound on r_c^d was obtained in [CIP09]: this bound is strictly less than 1. How do $\{s_k\}$ and $\{r_c^d\}$ relate? Santhanam and Srinivasan [SS12] answered this question by showing that $\{r_c^d\}$ is "interwoven" with $\{s_k\}$: for any numbers c and d, there is an integer k such that $r_c^d \le s_k$ and, similarly, in the converse direction. Therefore, $r_\infty^d = s_\infty$ for any d.

How important is a constant limit on the circuit depth in such "interweaving"? For example, how about Boolean formulas, a natural class of circuits of non-constant depth? Santhanam [San10], Seto and Tamaki [ST12] showed that satisfiability of linear-size Boolean formulas over the full binary basis can be tested exponentially faster than in $O(2^n)$ time: $f_c < 1$ for any constant c. This result suggests that we could expect f_c to be "interwoven" with $\{s_k\}$. In this paper, we prove this conjecture.

Our Results. It follows from the Sparsification Lemma [IPZ01] that for any positive integer k, there is a number c such that $s_k \le f_c$. We prove the "converse": for any number c, there is an integer k such that $f_c \le s_k$. Therefore, $f_\infty = s_\infty$. In fact, our main result is stronger in the following two aspects.

First, instead of $\{f_c\}$, we consider an analogous sequence of the exponential complexities for linear-size circuits of a more general type than Boolean formulas, see Section 4 for the definition of such circuits. Loosely speaking, a circuit of this type has two properties: (1) all gates have bounded fan-in and (2) each subcircuit has a bounded number of "border gates", i.e., gates that have wires to the "residual part" of the circuit. Any Boolean formula is a circuit of this type. Another special case is the class of circuits whose underlying graphs have bounded minimum vertex separation numbers [BFT09].

Second, we relate circuit satisfiability to k-CNF satisfiability using reductions that preserve satisfying assignments. More exactly, for any $\epsilon > 0$, a circuit ϕ with n variables is transformed in polynomial time into a k-CNF F such that

- F has the same variables as ϕ plus at most ϵn additional variables;
- if an assignment satisfies F then its restriction to the variables of ϕ satisfies ϕ; if an assignment satisfies ϕ, then it has a unique extension to the variables of F that satisfies F.

Thus, taking all assignments to the additional variables, ϕ can be transformed into an equivalent disjunction of subexponentially many k-CNFs. That is, any Boolean function computed by a Boolean formula can be computed by a disjunction of subexponentially many k-CNFs.

Note that the equality $f_\infty = s_\infty$ gives an equivalent statement for SETH: $f_\infty = 1$, see [CDL+12] for some other equivalent statements.

Organization of the Paper. The basic definitions and notation are given in Section 2. Section 3 describes how we reduce circuits to k-CNFs. The reducibility uses the extension rule (well known in proof complexity). Section 4 is about graphs underlying circuits for which we prove an "interweaving" with k-CNFs. In Section 5, we state and prove the main results.

2 Basic Definitions and Notation

Circuit Satisfiability. By a *circuit* we mean a single-output Boolean circuit in its most general version [Vol99] where, in particular, every input is labeled with either a variable or a truth value, but such a labeling is not necessarily one-to-one: distinct inputs may be labeled with the same variable. When talking about circuits, it will sometimes be convenient for us to use graph-theoretic terms instead of terms standard for circuits: for example, vertices and edges instead of nodes and wires, in-degree and out-degree instead of fan-in and fan-out, etc.

The number of nodes in a circuit ϕ is denoted by $|\phi|$. The set of variables labeling the inputs of ϕ is denoted by $var(\phi)$. Let A be an assignment of truth values to the variables of ϕ, i.e., a mapping from $var(\phi)$ to $\{0,1\}$. The value of ϕ on A is defined in the standard way [Vol99] and is denoted by $\phi(A)$. A circuit ϕ is called *satisfiable* if there is an assignment A such that $\phi(A) = 1$.

Let $\mathcal{C}$ be a set of circuits. The *satisfiability problem* for $\mathcal{C}$ is the problem of determining whether a circuit from $\mathcal{C}$ is satisfiable or not. We write $\mathcal{C}\ \mathsf{SAT}$ to denote the language consisting of all satisfiable circuits from $\mathcal{C}$.

Exponential Complexity. Let $\mathcal{C}$ be a class of circuits. Following Impagliazzo and Paturi [IP01] (see also [CP09] for details), we define the *exponential complexity* of $\mathcal{C}\ \mathsf{SAT}$ to be the infimum of constants α such that $\mathcal{C}\ \mathsf{SAT}$ can be decided by a two-sided error randomized algorithm in time $poly(|\phi|)\,2^{\alpha n}$ with probability of error $< \frac{1}{3}$:

$$exp\text{-}com(\mathcal{C}\ \mathsf{SAT}) = \inf\{\alpha \mid \mathcal{C}\ \mathsf{SAT} \in \mathbf{BPTIME}(poly(|\phi|)\,2^{\alpha n})\}.$$

Note that the polynomial $poly(|\phi|)$ in this definition may depend on α.

The fact that the definition above uses randomized algorithms, not deterministic ones, is not so important for this paper. All of our results remain valid if the exponential complexity of $\mathcal{C}$ SAT is defined as a similar measure where randomized algorithms are replaced by deterministic algorithms:

$$exp\text{-}com(\mathcal{C}\ \mathsf{SAT}) = \inf\{\alpha \mid \mathcal{C}\ \mathsf{SAT} \in \mathbf{DTIME}(poly(|\phi|)\, 2^{\alpha n})\}.$$

Boolean Formulas and CNFs. A *Boolean formula* is a circuit in which every gate has fan-in at most 2 and fan-out at most 1. Gates with fan-in 2 may be labeled with arbitrary binary Boolean functions. A *literal* is either a single-gate circuit or a two-gate circuit (where the output is labeled with the negation). A *clause* is either a literal or a circuit obtained from literals by adding wires from their outputs to a new output gate labeled with the disjunction of the corresponding arity. A *conjunctive normal form* (a *CNF* for short) is either a clause or a circuit obtained from clauses by adding wires from their outputs to a new output gate labeled with the conjunction of the corresponding arity. A CNF is called a *k-CNF* if every disjunction in its labeling has arity at most k.

3 Extension Rule for Circuits

In this section, we define circuit transformations based on the *extension rule*. This rule was introduced by Tseitin [Tse68] who used it to attain an exponential speed-up for the length of resolution proofs in propositional logic. A more general form of the rule is well known in proof complexity in connection with extended Frege systems, where the rule is used to abbreviate long formulas, see e.g. [Pud98]. In this general form, the extension rule allows using formulas of the form $z \leftrightarrow F$ in a proof, where F is any formula and z is a new propositional variable that appears neither in the previous part of the proof nor in the formula to be proved.

3.1 Induced Circuits

Let ϕ be a circuit and $G = (V, E)$ be its underlying directed graph. Each vertex $u \in V$ determines the directed graph $G_u = (V_u, E_u)$ where

$$V_u = \{u\} \cup \{w \in V \mid \text{there is a path from } w \text{ to } u \text{ in } G\};$$
$$E_u = \{(v, w) \in E \mid \text{both } v \text{ and } w \text{ are in } V_u\}.$$

This graph G_u is called the *subgraph induced by* u. A vertex $v \in V_u$ is said to be a *border vertex* of G_u if v has an outgoing edge incoming to a vertex outside G_u, i.e., there is an edge $(v, w) \in E$ where $w \in V - V_u$. The set of all border vertices of G_u is called the *border* of G_u and is denoted by $\beta(G_u)$. All other vertices in G_u are called *internal*.

The above terminology and notation are extended to circuits in a natural way: given ϕ and u, the *subcircuit induced by* u is the circuit whose underlying

graph is G_u and whose labeling is the same as in ϕ, i.e., for every vertex in V_u, its label in ϕ_u is the same as its label in ϕ. Note that the labeling for ϕ_u is defined correctly since for every vertex in V_u, its in-degree in G_u is the same as its in-degree in G (but its out-degrees in G_u and G may be different). The border of G_u is also referred as the *border* of ϕ_u and it is denoted by $\beta(\phi_u)$.

We want to "decompose" ϕ into two circuits: ϕ_u and a "residual" circuit obtained from ϕ by "contraction" of ϕ_u into a single vertex. This "residual" circuit and its underlying graph are denoted by $\phi \ominus \phi_u$ and $G \ominus G_u$ respectively. They are defined as follows:

- $G \ominus G_u$ is obtained from $G = (V, E)$ by removing all internal vertices of $G_u = (V_u, E_u)$ and removing all edges incident on these internal vertices. Thus, every vertex in $G \ominus G_u$ is either a vertex from $V - V_u$ or a border vertex of G_u.
- If a vertex belongs to $V - V_u$, its label in $\phi \ominus \phi_u$ is the same as in ϕ.
- If a vertex belongs to the border of G_u, it has in-degree 0 in $G \ominus G_u$. To label such border vertices, we use new variables, not occurring in ϕ. Namely, let $\beta(G_u) = \{v_1, \ldots, v_b\}$. We introduce b new variables $z_1, \ldots, z_b$ and we label each vertex v_i with z_i.

In this labeling of $v_1, \ldots, v_b$, each variable z_i "replaces" the subcircuit ϕ_{v_i} induced by v_i in ϕ. To emphasize this fact, we denote the circuit $\phi \ominus \phi_u$ using the standard notation for substitutions in formulas:

$$\phi[z_1/\phi_{v_1}, \ldots, z_b/\phi_{v_b}]. \tag{1}$$

It will be convenient for us to use either of the two notations: $\phi \ominus \phi_u$, which specifies the circuit up to names of new variables, or $\phi[z_1/\phi_{v_1}, \ldots, z_b/\phi_{v_b}]$, which specifies it completely.

3.2 Circuit Transformations

The extension rule is typically used to transform a formula F into an "equivalent" (in a special sense) formula $(z \leftrightarrow S) \wedge F[z/S]$ where S is a subformula of F. Here, we generalize this operation for circuits.

We begin with notation for composition of circuits, namely for circuits made up from other circuits using the Boolean functions $\leftrightarrow$ (equivalence) and $\wedge^m$ (m-ary conjunction). Given a circuit ϕ and a variable z not occurring in ϕ, the circuit denoted by $(z \leftrightarrow \phi)$ is obtained from ϕ by adding two new vertices and two new edges: a vertex v labeled with z, a vertex w labeled with the Boolean function $\leftrightarrow$, an edge from v to w, and an edge from the single output of ϕ to w. Thus, w is the output of the resulting circuit $z \leftrightarrow \phi$. Similarly, given circuits $\phi_1, \ldots, \phi_m$, the circuit denoted by $\phi_1 \wedge \ldots \wedge \phi_m$ is obtained by adding one new vertex and m new edges. The new vertex v is labeled with $\wedge^m$. The m new edges go from the outputs of $\phi_1, \ldots, \phi_m$ to v.

Let ϕ be a circuit and u be a vertex in ϕ. Consider the subcircuit ϕ_u induced by u, its underlying graph G_u, and the border of G_u. Let $\beta(G_u) = \{v_1, \ldots, v_b\}$

and let $z_1, \ldots, z_b$ be new variables not occurring in ϕ. We transform ϕ into the circuit

$$(z_1 \leftrightarrow \phi_{v_1}) \ \wedge \ \ldots \ \wedge \ (z_b \leftrightarrow \phi_{v_b}) \ \wedge \ \phi[z_1/\phi_{v_1}, \ldots, z_b/\phi_{v_b}] \tag{2}$$

and we write $\phi \overset{u}{\mapsto} \psi \wedge \phi'$ to denote this transformation, where ψ denotes the conjunction of the equivalences and ϕ' denotes the last conjunctive term in (2). The following simple lemma expresses the fact that such transformations preserve satisfiability.

Lemma 1. *Suppose that $\phi \overset{u}{\mapsto} \psi \wedge \phi'$. Then ϕ is satisfiable if and only if $\psi \wedge \phi'$ is satisfiable. Moreover,*

- *if an assignment satisfies $\psi \wedge \phi'$, then its restriction to $var(\phi)$ satisfies ϕ;*
- *if an assignment satisfies ϕ, then it has a unique extension to $var(\psi \wedge \phi')$ that satisfies $\psi \wedge \phi'$.*

Proof. It easily follows from the definition of $\phi[z_1/\phi_{v_1}, \ldots, z_b/\phi_{v_b}]$ that any satisfying assignment for (2) restricted to $var(\phi)$ satisfies ϕ. Conversely, any satisfying assignment A for ϕ can be extended to a satisfying assignment for circuit (2) by assigning values $\phi_{v_1}(A), \ldots, \phi_{v_b}(A)$ to the variables $z_1, \ldots, z_b$. Any other extension of A falsifies some of the equivalences in (2). □

3.3 Transformation Sequences

Our purpose is to transform a circuit into a conjunction of "small" circuits. A natural strategy is to apply successive transformations $\phi \overset{u}{\mapsto} \psi \wedge \phi'$ where u is chosen so that ϕ_u is a "small" subcircuit. That is, choose a vertex u_1 in ϕ such that ϕ_{u_1} is "small", then choose a vertex u_2 that induces a "small" subcircuit in $\phi \ominus \phi_{u_1}$, and so on. Below, we describe this approach in more precise terms.

Consider a circuit ϕ, a vertex u in ϕ, and the induced subcircuit ϕ_u. We call ϕ_u a (b, s)-*subcircuit* if $|\beta(\phi_u)| \leq b$ and $|\phi_u| \leq s$. A transformation $\phi \overset{u}{\mapsto} \psi \wedge \phi'$ is called a (b, s)-*transformation* if ϕ_u is a (b, s)-subcircuit.

Let $u_1, \ldots, u_l$ be a sequence of vertices in ϕ. We call it a (b, s)-*transformation sequence for* ϕ if there exist sequences $\{\phi_i\}_{i=0}^{l}$ and $\{\psi_i\}_{i=1}^{l}$ of circuits such that

- ϕ_0 is the circuit ϕ;
- for $i = 1, \ldots, l$,
  - u_i is a vertex in ϕ_{i-1};
  - there is (b, s)-transformation $\phi_{i-1} \overset{u_i}{\mapsto} \psi_i \wedge \phi_i$;
- ϕ_l has at most s vertices.

Lemma 2. *If a circuit ϕ has a (b, s)-transformation sequence σ of length l, then there is a circuit χ such that*

- *χ is a conjunction $\chi_1 \wedge \ldots \wedge \chi_t$, where $t \leq bl + 1$;*
- *each circuit χ_i has at most $s + 2$ vertices;*
- *$var(\phi) \subseteq var(\chi)$ and $|var(\chi)| \leq |var(\phi)| + bl$;*

– *ϕ is satisfiable if and only if χ is satisfiable.*

There is a polynomial-time algorithm that takes as input ϕ, σ and outputs a circuit χ that has the above properties.

Proof. The required algorithm takes ϕ, σ as input and constructs the sequence

$$\phi_0 \stackrel{u_1}{\mapsto} \psi_1 \wedge \phi_1, \quad \phi_1 \stackrel{u_2}{\mapsto} \psi_2 \wedge \phi_2, \quad \ldots, \quad \phi_{l-1} \stackrel{u_l}{\mapsto} \psi_l \wedge \phi_l$$

of (b, s)-transformations, where each ψ_i is a conjunction of b_i equivalences of the form $(z \leftrightarrow \phi_v)$. The number of such equivalences in the conjunction is equal to the number of vertices in the border of ϕ_i in ϕ_{i-1}. We denote this number by b_i and we write t to denote $\sum_{i=1}^{l} b_i + 1$. Next, the algorithm constructs the resulting circuit χ as the t-ary conjunction of circuits $\chi_1, \ldots, \chi_t$ where the first $t-1$ circuits $\chi_1, \ldots, \chi_{t-1}$ are equivalences of the form $(z \leftrightarrow \phi_v)$ and the last circuit χ_t is ϕ_l. Clearly, the algorithm constructs χ in polynomial time. We show that χ has the claimed properties.

By the definition of (b, s)-transformations, $b_i \leq b$ for $i = 1, \ldots, l$. Hence, we have $t \leq bl + 1$. Also, by the same definition, each circuit ϕ_v in an equivalence $(z \leftrightarrow \phi_v)$ has at most s vertices. Therefore, the equivalence itself has at most $s+2$ vertices. Since the last circuit χ_t has at most s vertices, each circuit χ_i has at most $s+2$ vertices. The number of new variables in χ is equal to the number of the equivalences, $\sum_{i=1}^{l} b_i$, which is at most bl. The remaining property (satisfiability preservation) is easily proved using Lemma 1 and induction on l. □

Corollary 1. *If a circuit ϕ has a (b, s)-transformation sequence σ of length l, then there is a k-CNF F such that*

– *$k \leq s + 2$;*
– *the number of clauses is at most $(bl + 1)2^k$;*
– *$var(\phi) \subseteq var(\chi)$ and $|var(\chi)| \leq |var(\phi)| + bl$;*
– *ϕ is satisfiable if and only if F is satisfiable.*

There is a polynomial-time algorithm that takes as input ϕ, σ and outputs a k-CNF F that has the above properties.

Proof. The circuit $\chi_1 \wedge \ldots \wedge \chi_t$ from Lemma 2 is transformed into a k-CNF F with the claimed properties as follows. Each circuit χ_i has at most $s + 2$ inputs and, therefore, it represents a Boolean function of at most $s + 2$ variables. Any such function can be computed by a k-CNF F_i where $k \leq s + 2$ and the number of clauses is not greater than 2^k. The k-CNF F is the conjunction $F_1 \wedge \ldots \wedge F_t$. □

Remark 1. According to the property of preserving satisfiability in Corollary 1, ϕ is satisfiable if and only if F is satisfiable. This equivalence is sufficient for the use of the corollary in Section 5. However, it is not difficult to see that a stronger form of this equivalence holds: each satisfying assignment for ϕ has a unique extension to $var(F)$ that satisfies F, and for each satisfying assignment for F, its restriction to $var(\phi)$ satisfies ϕ (cf. Lemma 1). The same applies to ϕ and χ in Lemma 2.

4 Graphs with (b, s)-Transformation Sequences

The notion of a (b, s)-transformation sequence is defined in terms of circuits (Section 3.3), but it is easy to see that, in fact, such sequences are determined by underlying graphs, independently of their labeling. Here is an equivalent definition in terms of graphs. Let ϕ be a circuit and $G = (V, E)$ be its underlying graph. Let σ be a sequence of vertices $u_1, \ldots, u_l$ in V. This sequence is a (b, s)-transformation sequence for ϕ if and only if there exist sequences $\{G_i\}_{i=0}^{l}$ and $\{H_i\}_{i=1}^{l}$ of graphs such that

- G_0 is the graph G;
- for $i = 1, \ldots, l$,
  - u_i is a vertex in G_{i-1} and H_i is the subgraph of G_{i-1} induced by u_i;
  - H_i has at most s vertices and the border of H_i in G_{i-1} consists of at most b vertices;
  - G_i is $G_{i-1} \ominus H_i$;
- G_l has at most s vertices.

Since σ is a (b, s)-transformation sequence for any circuit whose underlying graph is G, we refer to σ as a (b, s)-transformation sequence for G. In this section, we describe a class of graphs for which (b, s)-transformation sequences can be found in polynomial time.

The maximum in-degree in a graph G is denoted by *max-in*(G). The following lemma relates the maximum in-degree of graphs to sizes of induced subgraphs. Given a graph, does it have an induced subgraph with an "arbitrary" (in a reasonable sense) number of vertices?

Lemma 3. *Let $G = (V, E)$ be a graph with max-in$(G) \leq b$. For any integer t in the interval $b < t \leq |V|$, there is a vertex $u \in V$ such that the number of vertices of the induced subgraph $G_u = (V_u, E_u)$ is between t and $b(t-1)+1$:*

$$t \;\leq\; |V_u| \;\leq\; b(t-1)+1. \tag{3}$$

Proof. Induction on the depth of G. The basis step (the depth is 0, i.e., G is a single-vertex graph) is trivial. In the inductive step (the depth is positive), we select a vertex v that has two properties:

- v has non-zero in-degree;
- the subgraph G_v induced by v has at least t vertices.

Such a vertex exists, for example the sink of G has the above properties. Let d be the in-degree of v and let $v_1, \ldots, v_d$ be all in-neighbors of v. Consider the subgraphs $G_{v_1}, \ldots, G_{v_d}$ induced by $v_1, \ldots, v_d$ respectively and select an index k such that G_{v_k} has the maximum number of vertices among all these subgraphs. Let m be this maximum, the number of vertices in G_{v_k}.

There are only two options: either $t \leq m$ or $t > m$. If $t \leq m$, then the subgraph G_{v_k} has a required vertex u. This follows from the inductive assumption (the depth of G_{v_k} is less than the depth of G) and the fact that t does not exceed

the number of vertices in G_{v_k}. If $t > m$, then the vertex v can be taken as the vertex u required in the claim. Indeed, we have

$$t \;\leq\; \text{the number of vertices of } G_v \;\leq\; dm+1 \;\leq\; d(t-1)+1 \;\leq\; b(t-1)+1$$

and, thus, inequality (3) holds. □

For a graph $G = (V, E)$, the *maximum border size* of G is defined to be the maximum of $|\beta(G_u)|$ over all $u \in V$, i.e., the maximum border size of all induced subgraphs of G. We denote it by *max-border*(G).

Lemma 4. *For any graph $G = (V, E)$ with at least two vertices and for any numbers b and s such that*

$$\textit{max-in}(G) \leq b, \quad \textit{max-border}(G) \leq b, \quad 1 + b^2 < s \leq |V| \tag{4}$$

there exists a (b, s)-transformation sequence σ for G whose length is at most

$$\frac{b\,|V|}{s - b^2 - 1}. \tag{5}$$

There is a polynomial-time algorithm that takes as input G and numbers b, s satisfying all inequalities (4), and it outputs a (b, s)-transformation sequence σ for G with upper bound (5) on its length.

Proof. To construct a sequence σ and the corresponding sequences $\{G_i\}_{i=0}^{l}$ and $\{H_i\}_{i=1}^{l}$ of graphs (defined in the beginning of this section), we make l steps. At step i, we transform G_{i-1} into G_i by "cutting off" an induced subgraph H_i from G_{i-1}. A key point is that Lemma 3 can be used to find H_i such that the number of vertices of H_i is bounded from below and from above: this number lies between some t and s, where s given in the input and a value for t will be chosen later. Thus, on one hand, each subgraph H_i has at most s vertices, which is required for σ. On the other hand, when "cutting off" H_i, the number of vertices of G_{i-1} reduces by at least $t - b$. Hence, the total number of steps is at most $\lceil |V|/(t-b) \rceil$. It is clear that given t, this construction of σ takes polynomial time.

It remains to choose a value for t. By Lemma 3, for any t such that $b < t \leq |V|$, there is an induced subgraph with the number of vertices between t and $b(t-1)+1$. Therefore, an integer t must be chosen so as to satisfy

$$1 \leq b < t \;\text{ and }\; b(t-1)+1 \leq s.$$

We choose $t = \lceil (s-1)/b \rceil$, which guarantees that the second inequality above is satisfied for any $b \geq 1$ and $s \geq 1$. Then the first constraint holds if $b < (s-1)/b$. Thus, σ can be constructed for any b and s satisfying (4).

The length of σ is the number l of "cutting off" steps. Bound (5) on l follows from the fact that the number of steps does not exceed $|V|/(t-b)$. □

5 Exponential Complexity

In this section, we compare the exponential complexity of the satisfiability problems for certain classes of circuits. We begin with a definition of a suitable reducibility.

Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be classes of circuits. Let $\mathcal{C}_1$ SAT and $\mathcal{C}_2$ SAT be the languages consisting of satisfiable circuits from $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively. We say that $\mathcal{C}_1$ SAT is polynomial-time reducible to $\mathcal{C}_2$ SAT with *an arbitrarily small increase in the number of variables* if for every $\epsilon > 0$, there is a polynomial-time Karp reduction from $\mathcal{C}_1$ SAT to $\mathcal{C}_2$ SAT with the following additional property: for any circuit ϕ with n variables, R_ϵ maps ϕ into a circuit with at most $n + \epsilon n$ variables:

$$|var(R_\epsilon(\phi))| \ \leq \ n + \epsilon n.$$

Lemma 5. *If $\mathcal{C}_1$ SAT is polynomial-time reducible to $\mathcal{C}_2$ SAT with an arbitrarily small increase in the number of variables, then*

$$\textit{exp-com}(\mathcal{C}_1 \ \textit{SAT}) \ \leq \ \textit{exp-com}(\mathcal{C}_2 \ \textit{SAT}).$$

Proof. Suppose that $\mathcal{C}_1$ SAT is polynomial-time reducible to $\mathcal{C}_2$ SAT with an arbitrarily small increase in the number of variables. Also, suppose that there is an algorithm that solves $\mathcal{C}_1$ SAT in time $poly(|\phi|)\, 2^{\alpha n}$. Then the composition of this algorithm and the reduction gives an algorithm that solves $\mathcal{C}_2$ SAT in time $poly(|\phi|)\, 2^{\alpha(1+\epsilon)n}$. Taking $\epsilon \to 0$, we obtain the claim. □

We consider the satisfiability problems for the following classes of circuits:

- **k-CNFs.** For any $k \in \mathbb{N}$, the set of all k-CNFs is denoted by k-CNF. To denote the exponential complexity of k-CNF SAT, we use the same notation as in [IP01]:

$$s_k = \textit{exp-com}(k\text{-CNF SAT}) \quad \text{and} \quad s_\infty = \sup_k\{s_k\}.$$

- **Linear-Size Boolean Formulas.** For any number $c > 0$, let $\mathsf{FORMULA}_c$ denote the set of all Boolean formulas ϕ such that the number of vertices of ϕ is at most cn where $n = |var(\phi)|$. Recall that we consider Boolean formulas over the full basis (Section 2). The exponential complexity of $\mathsf{FORMULA}_c$ SAT is denoted using the following notation:

$$f_c = \textit{exp-com}(\mathsf{FORMULA}_c\ \mathsf{SAT}) \quad \text{and} \quad f_\infty = \sup_c\{f_c\}.$$

- **Linear-Size Circuits with Bounded fan-in and Bounded Border Size.** For any numbers $b \geq 1$ and $c > 0$, let $\mathsf{CIRCUIT}_{b,c}$ denote the set of all circuits ϕ such that
  - any node in ϕ has fan-in at most b;
  - the maximum border size of the graph underlying ϕ is at most b;
  - $|\phi| \leq cn$ where $n = |var(\phi)|$.

The exponential complexity of $\mathsf{CIRCUIT}_{b,c}\,\mathsf{SAT}$ is denoted using the following notation:

$$r_{b,c} = \mathit{exp\text{-}com}(\mathsf{CIRCUIT}_{b,c}\,\mathsf{SAT}) \quad \text{and} \quad r_{b,\infty} = \sup_c\{r_{b,c}\}.$$

Theorem 1. *For any $b \geq 1, c > 0$, there is an integer k such that $r_{b,c} \leq s_k$.*

Proof. Let $b \geq 1, c > 0, \epsilon > 0$ be fixed. We show that there is a polynomial-time algorithm that takes as input a circuit $\phi \in \mathsf{CIRCUIT}_{b,c}$ with n variables and outputs a k-CNF F such that

- $|\mathit{var}(F)| \leq n + \epsilon n$ for sufficiently large n;
- ϕ is satisfiable if and only if F is satisfiable.

By Lemma 5, the existence of such an algorithm implies $r_{b,c} \leq s_k$.

Take $s = \lceil 1 + b^2 + \frac{b^2 c}{\epsilon} \rceil$. By Lemma 4, if $n \geq s$, it takes polynomial time to construct a (b, s)-transformation sequence of length l for the graph underlying ϕ such that

$$l \leq \frac{b|\phi|}{s - b^2 - 1} \leq \frac{bcn}{s - b^2 - 1} \leq \frac{\epsilon n}{b}. \tag{6}$$

Next, by Corollary 1, it takes polynomial time to transform ϕ into a k-CNF F with $k \leq s + 2$ and $|\mathit{var}(F)| \leq n + bl$ such that ϕ is satisfiable if and only if F is satisfiable. Using inequality (6), we have $|\mathit{var}(F)| \leq n + \epsilon n$. □

Theorem 2. *For any integer $k \geq 1$, there is a number c such that $s_k \leq f_c$.*

Proof. It follows from the Sparsification Lemma [IPZ01] that satisfiability of k-CNFs has the same exponential complexity as satisfiability of linear-size k-CNFs. Since there is a trivial polynomial-time transformation of a k-CNF F with n variables and with $|F| \leq cn$ into an equivalent Boolean formula ϕ with the same variables and with $|\phi| \leq c'n$, the claim holds. □

Theorem 3. *For any $b \geq 2$ and $c > 0$, we have $f_c \leq r_{b,c}$.*

Proof. For any Boolean formula ϕ, the maximum fan-in is at most 2 and the maximum border size of the underlying graph is at most 1. Therefore, any Boolean formula ϕ with $|\phi| \leq cn$ is a circuit in $\mathsf{CIRCUIT}_{2,c}$. □

Theorem 4. *For any $b \geq 2$, we have $s_\infty = f_\infty = r_{b,\infty}$.*

Proof. Theorem 1 implies $r_{b,\infty} \leq s_\infty$, Theorem 2 implies $s_\infty \leq f_\infty$, and Theorem 3 implies $f_\infty \leq r_{b,\infty}$. □

References

[BFT09] Bodlaender, H.L., Fellows, M.R., Thilikos, D.M.: Derivation of algorithms for cutwidth and related graph layout parameters. Journal of Computer and System Sciences 75(4), 231–244 (2009)

[CDL+12] Cygan, M., Dell, H., Lokshtanov, D., Marx, D., Nederlof, J., Okamoto, Y., Paturi, R., Saurabh, S., Wahlström, M.: On problems as hard as CNF-Sat. In: Proceedings of the 27th Annual IEEE Conference on Computational Complexity, CCC 2012, pp. 74–84. IEEE Computer Society (2012)

[CIP09] Calabro, C., Impagliazzo, R., Paturi, R.: The complexity of satisfiability of small depth circuits. In: Chen, J., Fomin, F.V. (eds.) IWPEC 2009. LNCS, vol. 5917, pp. 75–85. Springer, Heidelberg (2009)

[CP09] Calabro, C., Paturi, R.: k-SAT is no harder than Decision-Unique-k-SAT. In: Frid, A., Morozov, A., Rybalchenko, A., Wagner, K.W. (eds.) CSR 2009. LNCS, vol. 5675, pp. 59–70. Springer, Heidelberg (2009)

[DH09] Dantsin, E., Hirsch, E.A.: Worst-case upper bounds. In: Handbook of Satisfiability, ch.12, pp. 403–424. IOS Press (2009)

[IP01] Impagliazzo, R., Paturi, R.: On the complexity of k-SAT. Journal of Computer and System Sciences 62(2), 367–375 (2001)

[IPZ01] Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity. Journal of Computer and System Sciences 63(4), 512–530 (2001)

[PP10] Paturi, R., Pudlák, P.: On the complexity of circuit satisfiability. In: Proceedings of the 42nd Annual ACM Symposium on Theory of Computing, STOC 2010, pp. 241–250. ACM (2010)

[Pud98] Pudlák, P.: The length of proofs. In: Buss, S.R. (ed.) Handbook of Proof Theory, pp. 547–637. Elsevier (1998)

[San10] Santhanam, R.: Fighting perebor: New and improved algorithms for formula and QBF satisfiability. In: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, pp. 183–192 (2010)

[SS12] Santhanam, R., Srinivasan, S.: On the Limits of Sparsification. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part I. LNCS, vol. 7391, pp. 774–785. Springer, Heidelberg (2012)

[ST12] Seto, K., Tamaki, S.: A satisfiability algorithm and average-case hardness for formulas over the full binary basis. In: Proceedings of the 27th Annual IEEE Conference on Computational Complexity, CCC 2012, pp. 107–116. IEEE Computer Society (2012)

[Tse68] Tseitin, G.S.: On the complexity of derivation in propositional calculus. In: Slisenko, A.O. (ed.) Studies in Constructive Mathematics and Mathematical Logic, Part II, pp. 115–125 (1968) (in Russian); Reprinted in: Siekmann, J., Wrightson, G. (eds.): Automation of Reasoning 2: Classical Papers on Computational Logic 1967-1970, pp. 466–483. Springer (1983)

[Vol99] Vollmer, H.: Introduction to Circuit Complexity: A Uniform Approach. Springer (1999)