A Satisfiability Algorithm for Depth-2 Circuits with a Symmetric Gate at the Top and AND Gates at the Bottom

Takayuki Sakai∗, Kazuhisa Seto†, Suguru Tamaki‡, Junichi Teruyama§

Abstract

In this paper, we present a moderately exponential time algorithm for the circuit satisfiability problem of depth-2 unbounded-fan-in circuits with an arbitrary symmetric gate at the top and AND gates at the bottom. As a special case, we obtain an algorithm for the maximum satisfiability problem that runs in time poly(n^t)·2^{n-\Omega(n)} for instances with n variables and O(n^t) clauses.

Key words: exponential time algorithm, maximum satisfiability, circuit satisfiability

∗Oki Electric Industry Co., Ltd.
†Seikei University seto@st.seikei.ac.jp. Supported in part by MEXT KAKENHI 24106003); JSPS KAKENHI (26730007).
‡Kyoto University tamak@kuis.kyoto-u.ac.jp. Supported in part by MEXT KAKENHI (24106003); JSPS KAKENHI (25240002; 26330011).
§National Institute of Informatics, and JST, ERATO, Kawarabayashi Large Graph Project teruyama@nii.ac.jp.
1 Introduction

1.1 Background

In the circuit satisfiability problem (Circuit SAT), our task is, given a Boolean circuit $C$, to decide whether there exists a 0/1 assignment to the input variables such that $C$ evaluates 1. If input instances are restricted to a class of Boolean circuits $C$, the problem is called $C$-SAT. A naive algorithm can solve Circuit SAT in time $O(\text{poly}(|C|) \cdot 2^n)$, where we denote by $|C|$ the size of $C$ and by $n$ the number of input variables of $C$ respectively. We say an algorithm for $C$-SAT is moderately exponential time if it checks the satisfiability of every $C \in C$ in time $\text{poly}(|C|) \cdot 2^{n-\omega(\log n)}$, i.e., super-polynomially faster than $2^n$. We are interested in for which class $C$ moderately exponential time satisfiability algorithms exist.

In this paper, we present a moderately exponential time algorithm for $\text{SYM} \circ \text{AND}$-SAT, where $\text{SYM} \circ \text{AND}$ is the class of depth-2 unbounded-fan-in circuits with an arbitrary symmetric gate at the top and AND gates at the bottom. Our algorithm can handle circuits of super-polynomial size. Such a result has not been known even for $\text{MAJ} \circ \text{AND}$-SAT that is a special case of $\text{SYM} \circ \text{AND}$-SAT, where symmetric gates are restricted to MAJORITY gates. $\text{MAJ} \circ \text{AND}$-SAT is equivalent to the maximum satisfiability problem (Max SAT)\(^1\) and usually treated as so in the context. Before stating our contribution more formally, we briefly survey Max SAT and Circuit SAT to explain our motivation for studying $\text{SYM} \circ \text{AND}$-SAT.

Maximum satisfiability

In Max SAT, the task is, given a set of clauses, to find a 0/1 assignment to the input variables that maximizes the number of satisfied clauses, where a clause is a disjunction of literals and a literal is a Boolean variable or its negation. Max SAT is one of the most fundamental NP-hard problems. In Max $k$-SAT, we pose a restriction on input instances that each clause contains at most $k$ literals. Max $k$-SAT is NP-hard even when $k = 2$.

Exponential time algorithms for Max SAT have been developed with respect to various parameters such as the number of variables, the number of clauses, the length of an instance and an objective value, see, e.g., [27] for the collection of previous results. With respect to the number of variables $n$, Williams gave an $O(2^{\omega n/3})$-time algorithm for Max 2-SAT [33], where $\omega < 2.3728639$ [19] is the exponent of the matrix multiplication. Since then, the existence of moderately exponential time algorithms for Max 3-SAT has been one of the major open questions in the study of exponential time algorithms, see, e.g., [5].

Circuit Satisfiability

Studying moderately exponential time algorithms for Circuit SAT is motivated by not only the importance in practice, e.g., logic circuit design and constraint satisfaction but also the viewpoint of Boolean circuit complexity. As pointed out by several papers such as [34, 38], there are strong connections between proving circuit lower bounds for $C$ and designing moderately exponential time algorithms for $C$-SAT.

Typical such connections are: (1) Some proof techniques such as deterministic/random restriction (shrinkage analysis/switching lemma) simultaneously prove circuit lower bounds for $C$ and give $C$-SAT algorithms, e.g., when $C$ is $\text{AC}^0$ circuits (bounded-depth unbounded-fan-in circuits with AND and OR gates) [2, 7, 15], or Boolean formulas [10, 11, 28, 32]. (2) As shown by

\(^1\)We do not distinguish $\text{MAJ} \circ \text{AND}$-SAT and $\text{MAJ} \circ \text{OR}$-SAT because we allow inputs to gates to be negated.
Williams [34, 37], if we obtain a moderately exponential time algorithm for \( C \)-SAT, then we also have a separation of complexity classes, i.e., \( \text{NEXP} \not\subseteq C \), where \( \text{NEXP} \) is the class of languages decidable by non-deterministic exponential time Turing machines.

These connections raise natural questions: (Q1) If we can prove circuit lower bounds for \( C \), then can we also obtain a moderately exponential time algorithm for \( C \)-SAT? (Q2) For which class of Boolean circuits \( C \) can we obtain a moderately exponential time satisfiability algorithm (so that we have \( \text{NEXP} \not\subseteq C \))? For both questions, one of the most interesting classes is \( \text{MAJ} \circ \text{AND} \). As for Q1, we know sub-exponential size circuits in \( \text{MAJ} \circ \text{AND} \) cannot compute the PARITY function [13]. As for Q2, Williams has shown moderately exponential time algorithms for so-called \( \text{ACC}^0 \circ \text{SYM} \) and \( \text{ACC}^0 \circ \text{THR} \) circuits and \( \text{NEXP} \not\subseteq \text{ACC}^0 \circ \text{SYM} \cup \text{ACC}^0 \circ \text{THR} \). The next natural target is \( \text{TC}^0 \), the class of bounded-depth unbounded-fan-in circuits with (weighted) linear threshold gates because \( \text{TC}^0 \) contains \( \text{ACC}^0 \circ \text{SYM} \) and \( \text{ACC}^0 \circ \text{THR} \). So far, moderately exponential time algorithms have been shown for a special case of \( \text{TC}^0 \) by Impagliazzo, Paturi and Schneider [17], where input instances are depth-2 circuits and have a linear number of wires. An obvious open question is to extend their result to handle polynomial size depth-2 circuits in \( \text{TC}^0 \). To do so, we must be able to handle polynomial size circuits in \( \text{MAJ} \circ \text{AND} \).

1.2 Our contribution

Our main result is the following theorem.

**Theorem 1.1.** We can count the number of satisfying assignments for \( C \in \text{SYM} \circ \text{AND}_k(n, m, w) \) deterministically in time

\[
\text{poly}(n, m, \log w) \cdot 2^{n - \Omega((n/\log(mw))^{\log n/4 \log(km)})}
\]

and exponential space.

Here we denote by \( \text{SYM} \circ \text{AND}_k(n, m, w) \) the class of circuits in \( \text{SYM} \circ \text{AND} \) with \( n \) variables and \( m \) AND gates of fan-in at most \( k \leq n \). Our algorithm can handle weighted symmetric gates and we denote by \( w \) the upper bound on the maximum weight of symmetric gates. See Section 2 and 3 for the formal definitions of weighted symmetric functions and \( \text{SYM} \circ \text{AND}_k(n, m, w) \). Our algorithm runs in time super-polynomially faster than \( 2^n \) when, e.g., \( m = n^{o(\log n/\log \log n)} \) and \( w = 2^{n^{0.99}} \). As a special case, we obtain an algorithm for the maximum satisfiability problem that runs in time \( \text{poly}(n^t) \cdot 2^{n-t^{O(1)}} \) for instances with \( n \) variables and \( O(n^t) \) clauses.

Although the running time of our algorithm is super-polynomially faster than \( 2^n \) instead of exponentially faster than \( 2^n \), this seems unavoidable due to the Strong Exponential Time Hypothesis (SETH) [7, 16, 18]: The hypothesis states that for all \( k \), there exists \( \varepsilon_k > 0 \) such that the satisfiability problem of \( k \)-CNF formulas cannot be solvable in time \( 2^{(1-\varepsilon_k)n} \). SETH has been used in proving conditional time lower bounds for several exponential time and polynomial time algorithms, see, e.g., [12, 20].

1.3 Related work

Moderately exponential time algorithms for \( C \)-SAT have been shown for various circuit classes \( C \) (sometimes with an additional condition on circuit size), e.g.,

- 3-CNF formulas \( \text{AND} \circ \text{OR}^3 \) [14, 21],
- \( k \)-CNF formulas \( \text{AND} \circ \text{OR}^k \) [22, 26, 30],
• CNF formulas \((\text{AND} \circ \text{OR})\) [6, 31],
• Symmetric Boolean Constraint Satisfaction Problems \((\text{AND} \circ \text{SYM})\) [1],
• Bounded Depth Circuits with AND/OR gates \((\text{AC}^0)\) [2, 7, 15],
• \(\text{AC}^0\) circuits with modulo gates \((\text{ACC}^0)\) [37],
• \(\text{ACC}^0\) circuits with symmetric/threshold gates at the bottom \((\text{ACC}^0 \circ \text{SYM} / \text{ACC}^0 \circ \text{THR})\) [36],
• weighted instances of Max SAT \((\text{THR} \circ \text{AND})\) [27],
• depth-2 threshold circuits \((\text{THR} \circ \text{THR})\) [17],
• De Morgan formulas [11, 28],
• formulas over the full binary basis [32] and finite bases [8],

• Boolean circuits [9, 24],

to name a few, see also [10, 23]. We remark that in [37], Williams gave an algorithm for \(\text{ACC}^0\)-SAT by combining a transformation from \(\text{ACC}^0\) circuits to \(\text{SYM} \circ \text{AND}\) circuits and a fast evaluation algorithm for \(\text{SYM} \circ \text{AND}\) circuits.

There are excellent surveys on connections between circuit lower bounds and algorithms for Circuit SAT and related topics [25, 29, 35].

1.4 Our techniques and paper organization

In Section 3, we give our first algorithm for \(\text{SYM} \circ \text{AND}_k\)-SAT based on dynamic programing. The algorithm is moderately exponential time when \(k\) is not too large, i.e., \(k = o(\log n / \log \log n)\).

In Section 4, we present our second algorithm for \(\text{SYM} \circ \text{AND}_k\)-SAT based on greedy restriction. The novelty of our algorithm and its analysis is a new way of reducing the bottom fan-in of circuits in a greedy manner. Intuitively, given a \(\text{SYM} \circ \text{AND}_k\) circuit with \(m\) gates, greedy restriction produces a collection of \(\text{SYM} \circ \text{AND}_{k'}\) circuits with \(k' = O(\log(km)/\log n)\) such that at least one of the circuits in the collection is satisfiable if and only if so is the original circuit. Note that previous techniques such as Schuler's width reduction [6, 31] or the standard random restriction achieve \(k' = O(\log(m/n))\) and the bound is not sufficient for our purpose.

Our bottom fan-in reduction is inspired by the similar techniques used in the context of Formula-SAT [10, 28, 32] and Max SAT [27] to reduce the size of instances. We show the efficiency of our bottom fan-in reduction and combine it with our first algorithm for \(\text{SYM} \circ \text{AND}_k\)-SAT.

2 Preliminaries

We use random access machines as our computation model.

Let \(V\) be the set \(\{x_1, \ldots, x_n\}\) of Boolean variables. A literal is either a variable or its negation. A term is a conjunction of literals. We use the value 1 to indicate Boolean ‘true’, and 0 ‘false’. A Boolean circuit \(C : \{0, 1\}^n \rightarrow \{0, 1\}\) is satisfiable if there exists a satisfying assignment for \(C\), i.e., an assignment \(a \in \{0, 1\}^n\) such that \(C(a) = 1\) holds.

A restriction is a mapping \(\rho : V \rightarrow \{0, 1, *\}\). The meaning of \(\rho\) is that if \(\rho(x_i) \in \{0, 1\}\), then we assign \(\rho(x_i)\) to \(x_i\), and if \(\rho(x_i) = *\), then we leave \(x_i\) as it is. Thus, when we apply a restriction
\( \rho \) to a Boolean function \( f \), we obtain the Boolean function \( f|_{\rho} \) defined over the variables \( \rho^{-1}(s) \). We also apply a restriction \( \rho \) to a Boolean circuit \( C \) and obtain a Boolean circuit \( C|_{\rho} \). When we apply a restriction \( \rho \) to a Boolean circuit \( C \), we simplify a Boolean circuit \( C \) using the standard transformation \( 0 \land f \equiv 0, 1 \land f \equiv f \) repeatedly. Here, for two Boolean functions (or circuits) \( f, g \) in the same variables, we write \( f \equiv g \) if \( f(a) = g(a) \) holds for all \( a \in \{0, 1\}^n \).

We denote by \( \mathbb{Z} \) the set of integers. A Boolean function \( f : \{0, 1\}^n \to \{0, 1\} \) is weighted symmetric if there exist a function \( g : \mathbb{Z} \to \{0, 1\} \) and integers \( w_0, w_1, \ldots, w_n \) such that \( f(x_1, \ldots, x_n) = g(w_0 + \sum_{i=1}^{n} w_i x_i) \) holds. In the rest of this paper, we assume that \( g(z) \) can be evaluated in time polynomial in \( \log_2 |z| \), where \( |z| \) denotes the absolute value of \( z \).

3 A Dynamic Programming Algorithm for \( \text{SYM} \circ \text{AND}_k \)

We denote by \( g \circ \text{AND}_k(n, m, w) \) the set of \( n \)-variable Boolean circuits of the form \( g(w_0 + \sum_{i=1}^{s} w_i t_i) \), where \( g : \mathbb{Z} \to \{0, 1\}, s \leq m, w_0, w_1, \ldots, w_s \in \mathbb{Z}, \max_{0 \leq i \leq s} |w_i| \leq w, \) and \( t_1, \ldots, t_s \) are terms that contain at most \( k \)-literals such that \( t_i \neq t_j \) holds for \( i \neq j \). We define

\[
\text{SYM} \circ \text{AND}_k(n, m, w) := \bigcup_{g: \mathbb{Z} \to \{0, 1\}} g \circ \text{AND}_k(n, m, w).
\]

We specify an element \( C \) in \( \text{SYM} \circ \text{AND}_k(n, m, w) \) as \( C = \{g, w_0, (t_1, w_1), \ldots, (t_s, w_s)\} \) and call \( s \) and \( \max_{0 \leq i \leq s} |w_i| \) the size and the maximum weight of \( C \) respectively.

For a restriction \( \rho \), we simplify \( C|_{\rho} = \{g, w_0, (t_1|_{\rho}, w_1), \ldots, (t_s|_{\rho}, w_s)\} \) repeatedly if there exists a pair \( (i, j), 1 \leq i < j \leq s \) such that \( t_i|_{\rho} \equiv t_j|_{\rho} \) holds. That is, we delete \( (t_j|_{\rho}, w_j) \) and replace \( (t_i|_{\rho}, w_i) \) by \( (t_i|_{\rho}, w_i + w_j) \).

Our first satisfiability algorithm for \( \text{SYM} \circ \text{AND}_k(n, m, w) \) is described in Fig. 1. The basic idea is as follows:

(Step 1) We construct a table \( T \) that contains pairs of the form \( (C, \#\text{sat}(C)) \) for every circuit \( C \) in \( g \circ \text{AND}_k(n', m', w') \), where \( \#\text{sat}(C) \) denotes the number of satisfying assignments for \( C \) and \( n', m', w' \) are appropriately chosen parameters. Furthermore, pairs are sorted in the lexicographical order with respect to the first coordinate \( C \) so that we can use binary search. To do so, we check the number of satisfying assignments for every circuit in \( g \circ \text{AND}_k(n', m', w') \) one by one in the lexicographical order using brute force search.

(Step 2) Let \( C \) be an input instance in \( g \circ \text{AND}_k(n, m, w) \). For each restriction \( \rho \) that assigns \( * \) to the first \( n' \) variables of \( C \), we check the number of satisfying assignments for \( C|_{\rho} \) using binary search in \( T \) and output the sum of them.

**Algorithm1** (\( C = \{g, w_0, (t_1, w_1), \ldots, (t_s, w_s)\} \): instance, \( n, m, k, w: \) integer)

1. if \( C \notin \text{SYM} \circ \text{AND}_k(n, m, w) \), return \( \perp \).
2. \( T \leftarrow \emptyset \). /* table for dynamic programming */
3. for each \( C \in g \circ \text{AND}_k(n', m', w') \), /* lexicographical order */
4. \( T \leftarrow T \cup \{(C, \#\text{sat}(C))\} \). /* brute force search */
5. \( N \leftarrow 0 \).
6. for each \( \rho : V \to \{0, 1,*\} \) such that \( \rho^{-1}(*) = \{x_1, \ldots, x_{n'}\} \),
7. \( N \leftarrow N + \#\text{sat}(C|_{\rho}) \). /* binary search in \( T \) */
8. return \( N \).

**Figure 1:** A Dynamic Programming Algorithm for \( \text{SYM} \circ \text{AND}_k \)
We will show the following theorem.

**Theorem 3.1.** We can count the number of satisfying assignments for $C \in \text{SYM} \circ \text{AND}_k(n, m, w)$ deterministically in time

$$\text{poly}(n, m, \log w) \cdot 2^{n-\Omega((n/\log(mw))^{1/k})},$$

and exponential space using Algorithm 1 for appropriately chosen $n', m'$.

**Proof.** We denote by $|g \circ \text{AND}_k(n, m, w)|$ the cardinality of $g \circ \text{AND}_k(n, m, w)$. To evaluate the running time of (Step 1), we upper bound the size of the table $T$ using the following lemma.

**Fact 3.2.** For all $m$, we have

$$|g \circ \text{AND}_k(n, m, w)| \leq (2w + 1)^k \sum_{i=0}^{k} 2^i \binom{n}{i} \leq 2^{(k+1)(2n)^k \log(2w+1)}.$$

**Proof.** Note that $\sum_{i=0}^{k} 2^i \binom{n}{i}$ is the number of different terms that consist of at most $k$-literals (including a constant function 1). Each term has a weight in $\{-w, -w+1, \ldots, w-1, w\}$. Thus, we have the first inequality. The second inequality follows from an elementary calculation.

Thus, we can bound the running time of Lines 03-04 from above by

$$2^{(k+1)(2n')^k \log(2(m+1)w+1)} \times \text{poly}(m', \log(mw)) \cdot 2^{n'},$$

where we set $n' = \sum_{i=0}^{k} 2^i \binom{n}{i} \leq (k+1)(2n)^k$.

Next we evaluate the running time of (Step 2). Note that the following guarantees that every $C|_\rho$ in Line 06 belongs to $g \circ \text{AND}_k(n', m', (m+1) \cdot w)$.

**Fact 3.3.** Let $C = \{g, w_0, (t_1, w_1), \ldots, (t_m, w_m)\}$. If $C \in g \circ \text{AND}_k(n, m, w)$ holds, then for all restriction $\rho$ with $|\rho^{-1}(*)| = n'$, we have $C|_\rho \in g \circ \text{AND}_k(n', m', (m+1) \cdot w)$.

**Proof.** By the definition of $\text{SYM} \circ \text{AND}_k(n, m, w)$, we have $\sum_{i=0}^{s} |w_i| \leq (m+1)w$. This implies the maximum weight of $C|_\rho$ is at most $(m+1)w$.

For each $C|_\rho$, binary search in Line 07 takes time at most

$$\log_2 |g \circ \text{AND}_k(n', m', (m+1) \cdot w)| \times \text{poly}(m', \log(mw)) = \text{poly}(m', \log(mw)).$$

Thus, we can bound the running time of Lines 06-07 above by

$$\text{poly}(m, m', \log(mw)) \cdot 2^{n-n'}.$$

If we set $n' = \left(\frac{n}{(k+1)^{2^{k+1} \log(2(m+1)w+1)}}\right)^{1/k} = \Theta((n/\log(mw))^{1/k})$, the total running time of Algorithm 1 is bounded from above by $\text{poly}(n, m, \log w) \cdot 2^{n-\Omega((n/\log(mw))^{1/k})}$. This completes the proof. 

This completes the proof.
4 A Greedy Restriction Algorithm for SYM ◦ AND_k

For a term $t$, we denote by $|t|$ the width of $t$, i.e., the number of literals in $t$ and by $\text{var}(t)$ the set of variables that appear in $t$ (possibly negated). Let $C \in \text{SYM} \circ \text{AND}_k(n, m, w)$ be a circuit \{ $g, w_0, (t_1, w_1), \ldots, (t_s, w_s)$ \}. We define $\text{var}_\ell(C) := \cup_{t_i \in t \geq \ell} \text{var}(t_i)$, $\text{freq}_\ell(C, x) := |\{t_i \in C \mid x \in \text{var}(t_i), |t_i| \geq \ell\}|$, and $L(C) := \sum_{t_i \in t \geq \ell} \|t_i\|$.

Our second satisfiability algorithm for $\text{SYM} \circ \text{AND}_k(n, m, w)$ is described in Fig. 2. The basic idea is as follows:
(Step 1) Choose a positive integer $\ell$ according to input. We seek for a variable, say $x$, that occurs most frequently in terms of width at least $\ell$. We recursively run the algorithm for $C|_{x=0}$ and $C|_{x=1}$. Here $C|_{x=a}$ denotes the circuit obtained from $C$ by applying a restriction $\rho$ such that $\rho(x) = a \in \{0, 1\}$ and $\rho(x') = *$ for $x' \neq x$.
(Step 2) If there is no term of width at least $\ell$, we call Algorithm 1.

Algorithm 2$(C = \{g, w_0, (t_1, w_1), \ldots, (t_s, w_s)\})$: instance, $n, n', \ell$: integer)
01: if $n > n'$,
02: \quad $x = \arg \max_{x \in \text{var}(C)} \text{freq}_\ell(C, x)$.
03: \quad $N_0 \leftarrow \text{Algorithm} 2(C|_{x=0}, n-1, \ell, \ell)$.
04: \quad $N_1 \leftarrow \text{Algorithm} 2(C|_{x=1}, n-1, \ell, \ell)$.
05: \quad return $N_0 + N_1$.
06: else
07: \quad $N \leftarrow 0$.
08: \quad for each $\rho : \text{var}(C) \rightarrow \{0, 1, \ast\}$ such that $\rho^{-1} (\{0, 1\}) = \text{var}_\ell(C)$,
09: \quad \quad $w' \leftarrow$ the maximum weight of $C|_{\rho}$.
10: \quad $N \leftarrow N + \text{Algorithm} 1(C|_{\rho}, n - |\text{var}_\ell(C)|, m', \ell - 1, w')$.
11: \quad return $N$.

Figure 2: A Greedy Restriction Algorithm for SYM ◦ AND_k

We will show the following theorem.

Theorem 4.1 (Restatement of Theorem 1.1). We can count the number of satisfying assignments for $C \in \text{SYM} \circ \text{AND}_k(n, m, w)$ deterministically in time
\[
\text{poly}(n, m, \log w) \cdot 2^{n - \Omega\left(\frac{n}{\log(mw)} \log n / 2 \log(km)\right)}
\]
and exponential space using Algorithm 2 for appropriately chosen $n', \ell, m'$.

Proof. Let us define a sequence of random variables $\{C_i\}$ inductively as $C_0 := C$ and $C_{i+1} := C_i|_{x=a}$, where $x = \arg \max_{x \in \text{var}(C_i)} \text{freq}_\ell(C_i, x)$ and $a$ is a uniform random bit.

We can think of the computation of Algorithm 2 as a rooted binary tree. That is, the root node is labeled with $C_0$, the left and right children of the root are labeled with $C_0|_{x=0}$ and $C_0|_{x=1}$, and so on. Then, if we pick a node of depth $n - n'$ uniformly at random, the distribution of its label is identical to that of the random variable $C_{n-n'}$.

We would like to bound the running time of Algorithm 2$(C_{n-n'}, n', \ell, \ell)$. It is obviously bounded from above by poly$(n, m, \log w) \cdot 2^n$. Furthermore, if $L_{\ell}(C_{n-n'}) < \frac{n'}{2}$ holds, the running time can be bounded by $2^{n'/2} \times$ (the running time of Algorithm 1$(C', n'/2, m', \ell - 1, w')$) for $C' \in \text{SYM} \circ \text{AND}_{\ell-1}(n'/2, m', w')$ with $m' = \ell \cdot (n')^{\ell-1}$ and $w' = (m + 1) \max_{0 \leq i \leq s} |w_i|$. We need the following lemma that is proven in the next section.
Lemma 4.2 (Greedy bottom fan-in reduction). Let $C \in \text{SYM} \circ \text{AND}_k(n, m, w)$. For all $n' \geq 4$, we have

$$\Pr \left[ L_\ell(C_{n-n'}) \geq 2^\ell \cdot L_\ell(C) \cdot \left( \frac{n'}{n} \right)^{\frac{\ell+2}{\ell}} \right] < 2^{-n'}.$$

Since $L_\ell(C) \leq km$, if we set $n' = \frac{1}{16} \left( \frac{n}{km} \right)^{2/\ell} \cdot n$ in the above lemma, we have

$$2^\ell \cdot L_\ell(C) \cdot \left( \frac{n'}{n} \right)^{\frac{\ell+2}{\ell}} \leq \frac{n'}{2},$$

that is, we have $L_\ell(C_{n-n'}) < n'/2$ with probability at least $1 - 2^{-n'}$. If we set $\ell = \frac{4 \log(km)}{\log n}$, then the total running time of Algorithm 2 is bounded from above by the sum of

$$\text{poly}(n, m, \log w) \cdot 2^{n-n'} \cdot 2^{-n'} \cdot 2^{n'}$$

and

$$\text{poly}(n, m, \log w) \cdot 2^{n-n'} \cdot (1 - 2^{-n'}) \cdot 2^{n'/2} \cdot 2^{n'/2 - \Omega((n'/(\log(m'w')))^{1/\ell})}$$

according to whether $L_\ell(C_{n-n'}) \geq n'/2$ holds or not. An elementary calculation completes the proof.

5 Proof of Lemma 4.2

The proof given here is essentially due to Chen, Kabanets, Kolokolova, Shaltiel and Zuckerman, see the proof of Lemma 4.3 in [10], except that we introduce $L_\ell(\cdot)$ and modify some parameters to measure the effect of bottom fan-in reduction rather than the shrinkage of De Morgan formulas.

Lemma 5.1 (Restatement of Lemma 4.2). Let $C \in \text{SYM} \circ \text{AND}_k(n, m, w)$. For all $n' \geq 4$, we have

$$\Pr \left[ L_\ell(C_{n-n'}) \geq 2^\ell \cdot L_\ell(C) \cdot \left( \frac{n'}{n} \right)^{\frac{\ell+2}{\ell}} \right] < 2^{-n'}.$$

We need the notion of super-martingales and a variant of Azuma’s inequality for them.

**Definition 5.2.** A sequence of random variables $X_0, X_1, \ldots, X_n$ is a super-martingale with respect to a sequence of random variables $Y_0, Y_1, \ldots, Y_n$ if it satisfies $E[X_i|Y_0, Y_1, \ldots, Y_{i-1}] \leq X_{i-1}$ for $1 \leq i \leq n$.

**Lemma 5.3** (Lemma 4.2 in [10]). Let $\{X_i\}_{i=0}^n$ be a super-martingale with respect to $\{Y_i\}_{i=0}^n$. Define $Z_i := X_i - X_{i-1}$ for $1 \leq i \leq n$. If, for $1 \leq i \leq n$, the random variable $Z_i$ (conditioned on $Y_0, Y_1, \ldots, Y_{i-1}$) takes two values with equal probability, and there exists a constant $c_i \geq 0$ such that $Z_i \leq c_i$ holds, then, for all positive real $\lambda$, we have

$$\Pr[X_n - X_0 \geq \lambda] \leq \exp \left( -\frac{\lambda^2}{2 \sum_{i=1}^n c_i^2} \right).$$

We begin with a lemma that estimates the effect of greedy restriction.
Lemma 5.4. Let $C \in \text{SYM} \circ \text{AND}_k(n, m, w)$ and $x = \arg \max_{x \in \text{var}(C)} \text{freq}_\ell(C, x)$. Then, we have

$$\max\{L_\ell(C|_{x=0}), L_\ell(C|_{x=1})\} \leq L_\ell(C) \cdot \left(1 - \frac{1}{n}\right)$$

and

$$\mathbb{E}_{a \in \{0, 1\}} [L_\ell(C|_{x=a})] \leq L_\ell(C) \cdot \left(1 - \frac{1}{n}\right) \frac{\ell + 2}{2}.$$

Proof. Pick any $t_i$ such that $|t_i| \geq \ell$ and $x \in \text{var}(t_i)$. If $|t_i| = \ell$, we have $|t_i|_{x=a} < \ell$ for all $a \in \{0, 1\}$. If $|t_i| > \ell$, we have $t_i|_{x=a} \equiv 0$ and $|t_i|_{x=a} = |t_i| - 1$ for some $a \in \{0, 1\}$. Since $\text{freq}_\ell(C, x) \geq \frac{L_\ell(C)}{n}$, we have $\max\{L_\ell(C|_{x=0}), L_\ell(C|_{x=1})\} \leq L_\ell(C) \cdot \left(1 - \frac{1}{n}\right)$ and

$$\mathbb{E}_{a \in \{0, 1\}} [L_\ell(C|_{x=a})] \leq L_\ell(C) - \frac{L_\ell(C)}{n} \min \left\{\ell, \left(\frac{1}{2} \cdot (\ell + 1) + \frac{1}{2} \cdot 1\right)\right\}$$

$$= L_\ell(C) \left(1 - \frac{\ell + 2}{2n}\right) \leq L_\ell(C) \cdot \left(1 - \frac{1}{n}\right) \frac{\ell + 2}{2}.$$

Recall that we define a sequence of random variables $\{C_i\}$ inductively as $C_0 := C$ and $C_{i+1} := C_i|_{x=a}$, where $x = \arg \max_{x \in \text{var}(C_i)} \text{freq}_\ell(C_i, x)$ and $a$ is a uniform random bit. We denote by $Y_i$ the random bit assigned to the selected variables in step $i$ for $1 \leq i \leq n$ and define $Y_0 := 0$. We define sequences of random variables $\{L_i\}_{i=0}^n, \{l_i\}_{i=0}^n, \{Z_i\}_{i=1}^n$ as follows: $L_i := L_\ell(C_i), l_i := \ln L_i$ and

$$Z_i := l_i - l_{i-1} - \frac{\ell + 2}{2} \ln \left(1 - \frac{1}{n - i + 1}\right).$$

Note that, given $Y_0, Y_1, \ldots, Y_{i-1}$, the random variable $Z_i$ takes two values with equal probability.

Lemma 5.5. Define $X_0 := 0$ and $X_i := \sum_{j=1}^{i} Z_j$. Then, the sequence of random variables $\{X_i\}_{i=0}^n$ is a super-martingale with respect to $\{Y_i\}_{i=0}^n$ and for each $Z_i$, we have $Z_i \leq c_i := -\frac{\ell}{2} \ln(1 - \frac{1}{n - i + 1})$.

Proof. By the first inequality of Lemma 5.4, we have $l_i \leq l_{i-1} + \ln \left(1 - \frac{1}{n - i + 1}\right)$. This implies

$$Z_i = l_i - l_{i-1} - \frac{\ell + 2}{2} \ln \left(1 - \frac{1}{n - i + 1}\right) \leq -\frac{\ell}{2} \ln \left(1 - \frac{1}{n - i + 1}\right) = c_i.$$ 

By Jensen’s inequality, we have $\mathbb{E}[l_i|Y_0, Y_1, \ldots, Y_{i-1}] \leq \ln \mathbb{E}[L_i|Y_0, Y_1, \ldots, Y_{i-1}]$. By the second inequality of Lemma 5.4, the right hand side is at most $\ln \left(L_{i-1} \cdot \left(1 - \frac{1}{n - i + 1}\right)^{\frac{\ell + 2}{2}}\right) = l_{i-1} + \frac{\ell + 2}{2} \ln \left(1 - \frac{1}{n - i + 1}\right)$. This implies $\mathbb{E}[Z_i|Y_0, Y_1, \ldots, Y_{i-1}] \leq 0$, that is, $\mathbb{E}[X_i|Y_0, Y_1, \ldots, Y_{i-1}] \leq \mathbb{E}[X_{i-1}|Y_0, Y_1, \ldots, Y_{i-1}] = X_{i-1}$. Thus, $\{X_i\}_{i=1}^n$ is a super-martingale.

Now we are ready to prove Lemma 5.1.

Proof of Lemma 5.1. Let $\lambda$ be arbitrary positive real and $c_i$’s be as defined in Lemma 5.5. By Lemma 5.3 and Lemma 5.5, we obtain

$$\Pr \left[ \sum_{j=1}^{i} Z_j \geq \lambda \right] \leq \exp \left( -\frac{\lambda^2}{2 \sum_{j=1}^{i} c_j^2} \right).$$
It is easy to show that $\sum_{j=1}^{i} Z_j = l_i - l_0 - \frac{\ell + 2}{2} \ln \frac{n-i}{n}$ by the definition of $Z_j$. Thus, we have

$$\Pr \left[ \sum_{j=1}^{i} Z_j \geq \lambda \right] = \Pr \left[ l_i - l_0 - \frac{\ell + 2}{2} \ln \left( \frac{n-i}{n} \right) \geq \lambda \right] = \Pr \left[ \mathcal{L}_i \geq e^\lambda \mathcal{L}_0 \left( \frac{n-i}{n} \right)^{\frac{\ell+2}{2}} \right].$$

For $1 \leq j \leq n-n'$, we have $c_j = -\frac{\ell}{2} \ln \left( 1 - \frac{1}{n-j+1} \right) \leq \frac{\ell}{2} \cdot \sqrt{\frac{2 \ln 2}{n-j+1}}$, using the inequality $-\ln(1-x) \leq \sqrt{\frac{2 \ln 2}{x}}$ for $0 < x \leq 1/4$. Thus, for $1 \leq i \leq n-n'$, $\sum_{j=1}^{i} c_j^2$ is at most

$$\frac{\ell^2 \ln 2}{2} \sum_{j=1}^{i} \left( \frac{1}{n-j+1} \right)^2 \leq \frac{\ell^2 \ln 2}{2} \sum_{j=1}^{i} \left( \frac{1}{n-j} - \frac{1}{n-j+1} \right) = \frac{\ell^2 \ln 2}{2} \left( \frac{1}{n-i} - \frac{1}{n} \right)$$

$$\leq \frac{\ell^2 \ln 2}{2} \cdot \frac{1}{n-i}.$$

Setting $i = n-n'$, we obtain

$$\Pr \left[ \mathcal{L}_{n-n'} \geq e^\lambda \mathcal{L}_0 \left( \frac{n'}{n} \right)^{\frac{\ell+2}{2}} \right] \leq \exp \left( -\frac{\lambda^2}{2 \sum_{j=1}^{n-n'} c_j^2} \right) \leq e^{-\frac{1}{\beta \ln 2} \lambda^2 n' \cdot n}. $$

Choosing $\lambda = \ell \ln 2$ completes the proof. \hfill $\Box$

### 6 Concluding Remarks

In this paper, we present a moderately exponential time algorithm for $\text{SYM} \circ \text{AND-SAT}$. We can extend our algorithm to handle bounded-depth unbounded-fan-in circuits with AND, OR and symmetric gates by combining the depth reduction algorithm due to Impagliazzo, Matthews and Paturi [15] and some transformation techniques due to Beigel, Reingold and Spielman [4] and Beigel [3]. The resulting algorithm runs in time super-polynomially faster than $2^n$ when the number of gates $m$ and the number of symmetric gates $t$ satisfy $mt = n \cdot \exp[o(\log n / \log \log n)]$ and $t = \exp[o(\log n / \log(mt/n))]$.

There are several interesting future directions. First, can we improve the upper bounds on the size of input instances for which moderately exponential time algorithms exist? We use a simple algorithm based on dynamic programming as the base algorithm and it might be improved by some sophisticated techniques in the design of exponential time algorithms. As for the bottom-fan-in reduction, it seems difficult to do better by only using greedy restriction.

Second, is it possible to give a moderately exponential time algorithm for $\text{THR} \circ \text{AND-SAT}$, where instances have polynomially many gates? Our algorithm can handle the case when the top gate is a linear threshold gate with weights of magnitude $2^{n^{0.99}}$. However, we need weights of magnitude $2^{\text{poly}(n)}$ to represent general linear threshold gates with polynomial number of inputs.
References


