Satisfiability Algorithm for Syntactic Read-k-times Branching Programs[*]

Atsuki Nagao[1], Kazuhisa Seto[2], and Junichi Teruyama[3]

1 **Seikei University, Tokyo, Japan**
a-nagao@st.seikei.ac.jp
2 **Seikei University, Tokyo, Japan**
seto@st.seikei.ac.jp
3 **National Institute of Informatics, and JST, ERATO, Kawarabayashi Large Graph Project, Tokyo, Japan**
teruyama@nii.ac.jp

Abstract

The satisfiability of a given branching program is to determine whether there exists a consistent path from the root to 1-sink. In a syntactic read-k-times branching program, each variable appears at most k times in any path from the root to a sink. We provide a satisfiability algorithm for syntactic read-k-times branching programs with n variables and m edges that runs in time $O\left(\text{poly}(n, m^{k^2}) \cdot 2^{(1-\mu(k))n}\right)$, where $\mu(k) = \frac{1}{4^{k+1}}$. Our algorithm is based on the decomposition technique shown by Borodin, Razborov and Smolensky [Computational Complexity, 1993].

1998 ACM Subject Classification F.2.0 Analysis of Algorithms and Problem Complexity: General

Keywords and phrases branching program, read-k-times, satisfiability, moderately exponential time, polynomial space

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2017.58

1 Introduction

Branching programs (BPs) are well studied computation models in theory and practice. A BP is a directed acyclic graph with a unique root node and two sink nodes. Each nonsink node is labeled using a variable, and the edges correspond to a variable's value of zero or one. Sink nodes are labeled either 0 or 1 depending on the output value. A BP computes a Boolean function naturally: it follows the edge corresponding to the input value from the root node to a sink node.

Given a BP, its satisfiability (BP SAT) involves the determination of whether there exists a consistent path from the root to 1-sink. Recently, BP SAT has become a significant problem because of the connection between satisfiability algorithms and lower bounds. Let C be a class of a circuit. Given a circuit in C, C-SAT is the determination of whether there exists an assignment to the input variables such that the circuit outputs 1. Williams [26] showed that to obtain $\mathbf{NEXP} \not\subseteq C$, it suffices to develop an $O\left(2^{n-\omega(\log n)}\right)$ time algorithm for C-SAT. Barrignton [3] showed that any function in $\mathbf{NC^1}$ can be computed using width-5 BPs of polynomial length. By combining these results, if we would like to prove $\mathbf{NEXP} \not\subseteq \mathbf{NC^1}$, it

[*] This work was supported in part by MEXT KAKENHI (24106003); JSPS KAKENHI (26730007); JST ERATO Grant Number JPMJER1201, Japan.

is sufficient to develop an $O\left(2^{n-\omega(\log n)}\right)$ time algorithm for width-5 BP SAT. In addition, the hardness of BP SAT implies the hardness of the Edit Distance and Longest Common Sequence problem [1]. Thus, the designing of a fast algorithm for BP SAT is one of the important tasks in the field of computational complexity.

For the SAT of some restricted BPs, polynomial or moderately exponential time algorithms are known. An ordered binary decision diagram (OBDD) is a BP that has the same order of variables in all paths from the root to any sink. By checking the reachability from the root to 1-sink, the OBDD SAT can be solved in polynomial time. A k-OBDD is a natural extension of an OBDD with k layers; all layers are OBDDs with the same order of variables. Bollig, Sauerhoff, Sieling, and Wegener [6] provided a polynomial time algorithm that solves the k-OBDD SAT for any constant k. A k-indexed binary decision diagram (k-IBDD) is the same as a k-OBDD, except that an OBDD in each layer may have a different order of variables. A k-IBDD SAT is known to be NP-complete when $k \geq 2$ [6]. Nagao, Seto, and Teruyama [18] proposed a satisfiability algorithm for any instances of k-IBDD SAT with cn edges, and its running time is $O\left(2^{(1-\mu_k(c))n}\right)$, where $\mu_k(c) = \Omega\left(\frac{1}{(\log c)^{2^{k-1}-1}}\right)$. Chen, Kabanets, Kolokolova, Shaltiel, and Zuckerman [10] showed that general BP SAT with $o(n^2)$ nodes can be determined in time $O\left(2^{n-\omega(\log n)}\right)$. However, there are not so much researches on BP SAT.

In this paper, we focus on syntactic read-k-times BPs. There exist two models of read-k-times BPs: *semantic* and *syntactic*. A read-k-times BP is *syntactic* if each variable appears at most k times in any path. It is *semantic* if each variable appears at most k times in any "computational" path. The semantic model is substantially stronger than the syntactic model. Beame, Saks, and Thathachar [5] showed that polynomial-size semantic read-twice BP can compute functions requiring exponential size on any syntactic read-k-times BP. To the best of our knowledge, non-trivial lower bounds on semantic read-twice BP are not known; however, the syntactic model is well-studied. Borodin, Razborov, and Smolensky [7] exhibited an explicit function of the lower bound of $\exp\left(\Omega\left(\frac{n}{k^3 4^k}\right)\right)$. Jukna [17] provided an explicit function f such that nondeterministic read-once BPs of polynomial size can compute $\neg f$ (i.e., the negation of f); however, to compute f, nondeterministic read-k-times BPs require a size of $\exp\left(\Omega\left(\frac{\sqrt{n}}{k^{2k}}\right)\right)$. Thathachar [25] showed that for any k, the computational power of read-$(k+1)$-times BPs is strictly stronger than that of read-k-times BPs. Sauerhoff [21] proved the exponential lower bound for randomized read-k-times BPs with a two-sided error.

When $k = 1$, syntactic read-k-times BP SAT can be determined in polynomial time by solving the reachability from the root to 1-sink. However, even when $k = 2$, this problem is known to be NP-complete; to the best our knowledge, there is no algorithm that is faster than the brute-force search. Therefore, we present a moderately exponential time algorithm for any constant $k \geq 2$. Our algorithm is based on the decomposition technique by Borodin, Razborov, and Smolensky [7].

▶ **Theorem 1.** *There exists a deterministic and polynomial space algorithm for a non-deterministic and syntactic read-k-times BP SAT with n variables and m edges that runs in time* $O\left(\mathrm{poly}(n, m^{k^2}) \cdot 2^{(1-4^{-k-1})n}\right)$.

1.1 Our Techniques

Our satisfiability algorithm consists of two steps as follows: **[Step 1: Decomposition]** Given a syntactic read-k-times BP B of m edges, we obtain the representation of a function computed by B as a disjunction of at most m^{2k^2} decomposed functions by using the decomposition algorithm proposed by Borodin, Razborov, and Smolensky [7]. It is sufficient to check the satisfiability of each decomposed function in the running time of Theorem 1, because if one of these functions is satisfiable then the input B is also satisfiable. Moreover, the property of the decomposition algorithm states that each decomposed function is a conjunction of at most $2k^2$ functions on small variable sets. Let us represent a conjunction of functions as a set of functions $\mathcal{F} = \{f_1, f_2, \dots, f_\ell\}$, where $\ell \leq 2k^2$. **[Step 2: Satisfiability Checking]** To check the satisfiability of $\mathcal{F}$, we find an assignment that all functions f_i are satisfied at the same time. Let $(\mathcal{F}_1, \mathcal{F}_2)$ be a partition of $\mathcal{F}$. In addition, let X_1 and X_2 be sets of input variables appearing in only $\mathcal{F}_1$ and $\mathcal{F}_2$ respectively and X_3 be a set of input variables appearing in both F_1 and F_2. If X_3 is an empty set, we can check the satisfiability of $\mathcal{F}_1$ and $\mathcal{F}_2$ independently in time $O(2^{|X_1|} + 2^{|X_2|})$ by exhaustive search on each set X_1 and X_2. If both $\mathcal{F}_1$ and $\mathcal{F}_2$ are satisfiable, we know that $\mathcal{F}$ is also satisfiable. Our algorithm assigns 0/1 value to the variables in X_3 and then performs the exhaustive search on each set X_1 and X_2. Assuming that $|X_1| + |X_2| + |X_3| = n$, we obtain the satisfiability of $\mathcal{F}$ in time $O(2^{|X_3|}(2^{|X_1|} + 2^{|X_2|})) = O(2^{n-\min\{|X_1|,|X_2|\}})$. Further, using probabilistic method, we show that the existence of a partition $(\mathcal{F}_1, \mathcal{F}_2)$ of $\mathcal{F}$ such that the value $\min\{|X_1|, |X_2|\}$ is adequately large to imply the running time in Theorem 1. Thus, we can save the running time of our satisfiability algorithm.

1.2 Related Work

A circuit satisfiability problem is, given a Boolean circuit, to find an assignment to the inputs of the circuit such that the circuit outputs 1. Recently, this problem has been studied extensively, and excellent algorithms that can outperform a brute-force search have been known for some restricted circuit classes such as conjunctive normal forms [2, 8, 12, 13, 14, 19, 22], $\mathbf{AC}^0$ [4, 9, 15], $\mathbf{ACC}^0$ [27], depth-2 threshold circuits [16], De Morgan formulas [11, 20, 24], and formulas over the full binary basis [23].

Paper Organization

The remainder of this paper is organized as follows. In Section 2, we provide the notation and definitions. In Section 3, we provide two algorithms. One is a decomposition algorithm based on the technique in [7]. The other is a satisfiability algorithm for a specific class of Boolean functions. In Section 4, we propose our satisfiability algorithm for syntactic read-k-times BPs.

2 Preliminaries

A set of integers $\{1, 2, \dots, n\}$ is denoted by $[n]$. For a set S, $|S|$ denotes the cardinality of S. Let $X = \{x_1, \dots, x_n\}$ be a set of Boolean variables, and for $x \in X$, $\overline{x}$ denotes the negation of x. A *branching program* (*BP*), denoted by $B = (V, E)$, is a rooted directed acyclic multigraph. A BP has a unique root node r and two sink nodes (0-sink and 1-sink); 0-sink and 1-sink are nodes labeled by **0** and **1**, respectively. Each node except for the sink nodes is labeled from X. Each edge $e \in E$ has a label 0 (*0-edge*) or 1 (*1-edge*). We call node v an x_i*-node* when v's label is x_i. A BP B is *deterministic* if any nodes except the two sink nodes in B have exactly two outgoing edges: one is a 0-edge, and the other is a 1-edge. Otherwise, B is *nondeterministic*. For an edge $e = (u, v) \in E$, u is a *parent* of v and the *head* of e. The *in-degree* of v is defined as the number of parents of v.

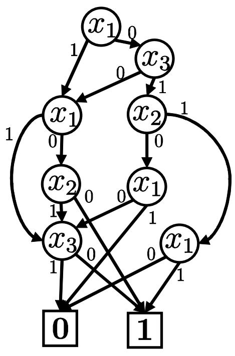

Figure 1 Syntactic read-twice branching program.

For a BP B on X, each input $\alpha = (\alpha_1, \ldots, \alpha_n) \in \{0,1\}^n$ activates all α_i-edges leaving the x_i-nodes in B, where $1 \leq i \leq n$. A *computation path* is a path from r to a 0-sink or from r to a 1-sink using only activated edges. A BP B outputs 0 if there is no computation path from the root r to a 1-sink; otherwise, B outputs 1. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. A BP B *represents* f if $f(\alpha)$ is equal to the output of B for any assignment $\alpha \in \{0,1\}^n$. Two BPs B_1 and B_2 are *equivalent* if B_1 and B_2 represent the same function. The size of B, denoted by $|B|$, is defined as the number of edges in B. A BP is *syntactic read-k-times* if each variable appears at most k times in each path. Figure 1 is an example of syntactic read-twice BPs ($k = 2$). A BP is *semantic read-k-times* if each variable appears at most k times in each computation path. In this paper, we use only the syntactic model and for simplicity we call it read-k-times BP.

For a BP B and two nodes v, w, a *subbranching program* $\langle B, v, w\rangle$ is a BP that contains v as the root node, w as the sink node, and every nodes and edges in all v-w paths in B. Given a BP B and nodes $v, w \in V$, $\langle B, v, w\rangle$ is constructed as follows:

1. Let V' be the subset of V such that $u \in V'$ is reachable from v and to w.
2. Output the subgraph of B induced by V'.

Note that, for any pair of nodes v and w, we can construct $\langle B, v, w\rangle$ in $O(|B|)$.

A *partial assignment* to $x = (x_1, \ldots, x_n)$ is $\alpha = (\alpha_1, \ldots, \alpha_n) \in \{0,1,*\}^n$ such that x_i is unset when $\alpha_i = *$; otherwise x_i is assigned to α_i. For any partial assignment $\alpha \in \{0,1,*\}^n$, a *support* of α is defined as $S(\alpha) := \{x_i \mid \alpha_i \neq *\}$. For partial assignments α and α' such that $S(\alpha)$ and $S(\alpha')$ are disjoint, $\alpha \circ \alpha'$ denotes the *composition* of α and α': $\alpha \circ \alpha'(i) = \alpha(i)$ if $x_i \in S(\alpha)$, $\alpha \circ \alpha'(i) = \alpha'(i)$ if $x_i \in S(\alpha')$, and $\alpha \circ \alpha'(i) = *$ otherwise. For instance, when $\alpha = (1, *, *)$ and $\alpha' = (*, *, 0)$, $\alpha \circ \alpha' = (1, *, 0)$.

3 Key Lemmas

In this section, we provide two key lemmas for our algorithm. First, we introduce the decomposition algorithm developed by Borodin, Razborov, and Smolensky [7]. Their algorithm decomposes a (nondeterministic) read-k-times BP into a set of BPs with a small number of variables. Next, we provide a satisfiability algorithm for a specific class of Boolean functions that have three properties with parameters a and k: (1) Each function is composed of a disjunction of ka subfunctions. (2) Each variable belongs to at most k subfunctions. (3) Each subfunction has at most n/a variables. Our algorithm that checks the satisfiability of such a function is exponentially faster than a brute-force search.

Now, we analyze the running time of a decomposition algorithm by Borodin, Razborov, and Smolensky [7]. We will use this algorithm as a module to solve the syntactic read-k-times BP SAT in Section 4.1.

▶ **Lemma 2** (Theorem 1 in [7]). *Let B be a (nondeterministic) syntactic read-k-times BP with n variables and size m and represent a Boolean function $f : \{0,1\}^n \to \{0,1\}$, and let a be a positive integer. There is an algorithm that constructs kam^{ka} BPs $\{B_{i,j}\}$ from B, where $i \in [m^{ka}]$ and $j \in [ka]$, such that the following properties hold:*

1. *Let $f_{i,j}$ be the Boolean function represented by $B_{i,j}$. Then,*
$$f = \bigvee_{i\in[m^{ka}]} \bigwedge_{j\in[ka]} f_{i,j}.$$
2. *Let $X_{i,j}$ be the set of variables that appear in $B_{i,j}$. For each i and j, $|X_{i,j}|$ is at most $\lceil n/a \rceil$. For each i, each variable x belongs to at most k sets of $\{X_{i,j}\}_{j=1,\dots,ka}$.*

▶ **Lemma 3.** *Given a (nondeterministic) syntactic read-k-times BP B with n variables and size m, the running time of algorithm in Lemma 2 is at most $O(kam^{ka+1})$.*

Proof. Let us observe the construction given in the proof of Theorem 1 in [7]. Let B be a nondeterministic and syntactic read-k-times BP with n variables and size m. For each pair of nodes $(v,w) \in V^2$, $X(v,w)$ denotes the set of all variables that appear in the labels on all possible paths from v to w except for the label of w.

We call a sequence $e_1 := (w_1, v_2), e_2 := (w_2, v_3), \dots, e_\ell := (w_\ell, v_{\ell+1})$ of edges a *trace* if and only if the following properties hold:

(a) For each j with $1 \le j \le \ell+1$, we have $|X(v_j, w_j)| < n/a$.
(b) For each j with $1 \le j \le \ell$, we have $|X(v_j, v_{j+1})| \ge n/a$,

where we set v_1 as the root and $w_{\ell+1}$ as the 1-sink.

Note that any path from r to the 1-sink contains a unique trace. Let $\mathcal{T}$ be the set of all traces. For each trace $T = (e_1 = (w_1, v_2), \dots, e_\ell = (w_\ell, v_{\ell+1})) \in \mathcal{T}$ and $1 \le j \le \ell$, let $B_{T,j}$ be a BP constructed as follows:

1. Prepare the subbranching program $\langle B, v_j, w_j \rangle$, 0-sink, and 1-sink.
2. Create an edge from w_j to the 1-sink with the same label of (w_j, v_{j+1}).
3. If some node v does not have a 0-edge (resp. 1-edge) as an outgoing edge, create a 0-edge (resp. 1-edge) from v to the 0-sink.

Intuitively, $B_{T,j}$ contains all paths from v_j to v_{j+1} through w_j. Note that the index i of the statement corresponds to each trace T. Let $g_{T,j}$ be the function represented by $B_{T,j}$. Then, we have $f = \bigvee_{T\in\mathcal{T}} \bigwedge_{j=1}^{\ell+1} g_{T,j}$. Each function $g_{T,j}$ depends on at most $\lceil n/a \rceil$ variables by property (a). Because B is a syntactic read-k-times BP, for each trace T and each variable x, at most k functions $g_{T,j}$ depend on x. By property (b), we have

$$\sum_{j=1}^{\ell} |X(v_j, v_{j+1})| + |X(v_{\ell+1}, w_{\ell+1})| \ge \frac{n\ell}{a} + |X(v_{\ell+1}, w_{\ell+1})|,$$

where $w_{\ell+1}$ is the 1-sink. Since each variable belongs to at most k subbranching programs, the left-hand side can be bounded above by kn. Then, $\ell \le ka$ holds. Moreover, $\ell = ka$ holds only if $|X(v_{\ell+1}, w_{\ell+1})| = 0$, in which case $g_{T,\ell+1}$ is a constant function. If this constant is 0, then $\bigwedge_{j=1}^{\ell+1} g_{T,j}$ is equal to 0 and we can drop whole terms. If it is 1, we can drop $g_{T,\ell+1}$. Therefore, each conjunction part consists of at most ka terms. The number of traces $|\mathcal{T}|$ is at most m^{ka} because $\ell \le ka$ holds.

The rest of the proof is to analyze the running time of the above construction. First, we find $X(v,w)$ by dynamic programming in $O(m)$ time for each pair of nodes $v, w \in$

V. Then, the running time for enumerating all $X(v,w)$ is at most $O(m^3)$. Using the database of $X(v,w)$, we enumerate all traces by a DFS-like search. The running time for enumerating all traces is at most $O(mka \cdot |\mathcal{T}|)$. For each trace $T \in \mathcal{T}$, we construct at most ka branching programs $B_{t,j}$ in $O(mka)$ time. Then, the total running time is at most $O(m^3) + O(mka \cdot |\mathcal{T}|) = O(kam^{ka+1})$. ◀

Next, we prove the following lemma for the satisfiability algorithm for a specific class of Boolean functions.

▶ **Lemma 4.** *Let a and k be positive integers with $a \le n$. Suppose that we are given a set of ka functions f_i that satisfy the following properties:*
1. *Each f_i depends on only at most $\lceil n/a \rceil$ variables $X_i \subset X$, i.e., $|X_i| \le \lceil n/a \rceil$.*
2. *Each variable x belongs to at most k sets X_i.*
3. *Each function f_i can be computed in a time of at most t and a space of at most s.*

Then, there exists a deterministic algorithm for counting the satisfiable assignments of the function $f = \bigwedge_i f_i$ that runs in time $O\left(2^{ka}kan\right) + O(kat) \cdot 2^{\left(1-\frac{2}{4^{k+1}}\left(1-\frac{k}{a}\right)\right)n}$ and space $O(s+kan)$.

Proof. Suppose that n is even. (In the case when n is odd, we also obtain the same result in a similar way.) We can also assume that each variable x belongs to at least one set X_i. If all sets X_i do not contain a variable x, then $\sum_i |X_i| \le k(n-1)$. This implies that there exists a set X_i such that $|X_i| \le (n-1)/a < \lceil n/a \rceil$. Then, we can put the variable x into the set X_i while preserving the properties.

Let $\mathcal{F}$ be the family of all subsets of $[ka]$. The size of $\mathcal{F}$, i.e., $|\mathcal{F}|$ is 2^{ka}. For $F \in \mathcal{F}$, $\bar{F}$ is defined as $[ka] \setminus F$. We define the set of variables $V_F := \left(\bigcup_{i\in F} X_i\right) \setminus \left(\bigcup_{i \in \bar{F}} X_i\right)$. The set V_F contains all variables that belong to only $\bigcup_{i \in F} X_i$. By definition, for any $F \in \mathcal{F}$, V_F and $V_{\bar{F}}$ are disjoint.

Find the set $F \in \mathcal{F}$ that maximizes $\min\{|V_F|, |V_{\bar{F}}|\}$ in $|\mathcal{F}| \cdot O(kan) = O(2^{ka}kan)$ time by an exhaustive search for $\mathcal{F}$. Let $Y := \{x_1, \dots, x_n\} \setminus (V_F \cup V_{\bar{F}})$. Apply some partial assignment α whose support is Y. Then, all $f_i|_\alpha$ for $i \in F$ (resp. $i \in \bar{F}$) depend on only the variables in V_F (resp. $V_{\bar{F}}$). Let A_F be a set of partial assignments α_F such that $S(\alpha_F) = V_F$, and $f_i|_\alpha(\alpha_F) = 1$ holds for all $i \in F$. Similarly, let $A_{\bar{F}}$ be a set of partial assignments $\alpha_{\bar{F}}$ such that $S(\alpha_{\bar{F}}) = V_{\bar{F}}$, and $f_i|_\alpha(\alpha_{\bar{F}}) = 1$ holds for all $i \in \bar{F}$. By an exhaustive search for all partial assignments whose support is V_F (resp. $V_{\bar{F}}$), we count the number of elements of A_F (resp. $A_{\bar{F}}$). Since $f = \bigwedge_{i=1}^{ka} f_i$, for $\alpha_F \in A_F$ and $\alpha_{\bar{F}} \in A_{\bar{F}}$, $f(\alpha \circ \alpha_F \circ \alpha_{\bar{F}}) = 1$ holds. Then, the number of assignments that satisfy f and contain a partial assignment α is $|A_F| \cdot |A_{\bar{F}}|$.

We can count the satisfiable assignments of f by the above operations for all partial assignments α where $S(\alpha) = Y$. For each i, the number of times for computing the function f_i is at most $2^{|Y|} \cdot 2^{\max\{|V_F|,|V_{\bar{F}}|\}}$. Using $|Y| + |V_F| + |V_{\bar{F}}| = n$, we have $2^{|Y|} \cdot 2^{\max\{|V_F|,|V_{\bar{F}}|\}} = 2^{n-\min\{|V_F|,|V_{\bar{F}}|\}}$. Thus, the running time is at most

$$O(2^{ka}kan) + kat \cdot 2^{n-\min\{|V_F|,|V_{\bar{F}}|\}}.$$

Now, we show that $\max_{F\in\mathcal{F}} \min\{|V_F|, |V_{\bar{F}}|\}$ is at least $\frac{2}{4^{k+1}}\left(1-\frac{k}{a}\right)n - \frac{1}{2}$. It follows that the running time of our algorithm is

$$\begin{aligned} O(2^{ka}kan) + kat \cdot 2^{n-\min\{|V_F|,|V_{\bar{F}}|\}} &\le O\left(2^{ka}kan\right) + kat \cdot 2^{n-\frac{2}{4^{k+1}}\left(1-\frac{k}{a}\right)n+\frac{1}{2}} \\ &= O\left(2^{ka}kan\right) + \sqrt{2}kat \cdot 2^{\left(1-\frac{2}{4^{k+1}}\left(1-\frac{k}{a}\right)\right)n}. \end{aligned}$$

Let S be the set of variables $\{x_1, \dots, x_{n/2}\}$ and L be the set of variables $\{x_{(n/2)+1}, \dots, x_n\}$. Now, we define good/bad pairs of variables. This notation is used in the proof of Theorem 6 in [7]. A pair $(x, x') \in S \times L$ is *good* iff there is no $i \in [ka]$ such that $x \in X_i$ and $x' \in X_i$ and *bad* otherwise. For each i, the number of bad pairs for X_i is $|S \cap X_i| \cdot |L \cap X_i|$. Since $|S \cap X_i| + |L \cap X_i| = |X_i| \leq \lceil \frac{n}{a} \rceil$ hold, we have

$$|S \cap X_i| \cdot |L \cap X_i| \leq \frac{1}{4} \cdot \left\lceil \frac{n}{a} \right\rceil^2 .$$

By summing the number of bad pairs for all X_i, the total number of bad pairs is at most $\frac{ka}{4} \cdot \lceil \frac{n}{a} \rceil^2$. Then, using $\lceil \frac{n}{a} \rceil < \frac{n}{a} + 1$ and $a \leq n$, the number of good pairs is at least

$$\frac{n^2}{4} - \frac{ka}{4} \cdot \left\lceil \frac{n}{a} \right\rceil^2 > \frac{1}{4}\left(1 - \frac{k}{a}\right) n^2 - \frac{3k}{4} n.$$

Let us consider that the set $F \in \mathcal{F}$ is chosen uniformly at random. For each good pair $(x, x') \in S \times L$, $\mathbf{Pr}[x \in V_F, x' \in V_{\bar{F}}] \geq 4^{-k}$. Hence,

$$\mathop{\mathbf{E}}_{F \in \mathcal{F}}[|\{(x, x') \mid x \in S \cap V_F, x' \in L \cap V_{\bar{F}}\}|] \geq \frac{1}{4^k}\left[\frac{1}{4}\left(1 - \frac{k}{a}\right) n^2 - \frac{3kn}{4}\right]$$

holds. This implies that there exists a set F such that

$$|V_F| \cdot |V_{\bar{F}}| \geq |S \cap V_F| \cdot |L \cap V_{\bar{F}}| \geq \frac{1}{4^k}\left[\frac{1}{4}\left(1 - \frac{k}{a}\right) n^2 - \frac{3kn}{4}\right].$$

For such a set $F \in \mathcal{F}$, if $|S \cap V_F| \cdot |L \cap V_{\bar{F}}| \geq M$ for some value M, then we have

$$\min\{|S \cap V_F|, |L \cap V_{\bar{F}}|\} \geq \frac{M}{\max\{|S \cap V_F|, |L \cap V_{\bar{F}}|\}} \geq \frac{2M}{n}.$$

We used the fact that $|S \cap V_F|, |L \cap V_{\bar{F}}| \leq \frac{n}{2}$. Since $\min\{|V_F|, |V_{\bar{F}}|\} \geq \min\{|S \cap V_F|, |L \cap V_{\bar{F}}|\}$ holds and n and k are nonnegative integers, we have

$$\begin{aligned}
\min\{|V_F|, |V_{\bar{F}}|\} &\geq \frac{2}{4^k n}\left[\frac{1}{4}\left(1 - \frac{k}{a}\right) n^2 - \frac{3kn}{4}\right] \\
&= \frac{2}{4^{k+1}}\left(1 - \frac{k}{a}\right) n - \frac{6k}{4^{k+1}} \\
&> \frac{2}{4^{k+1}}\left(1 - \frac{k}{a}\right) n - \frac{1}{2}.
\end{aligned}$$

The last inequality is by the fact that for any $k \geq 1$, $\frac{6k}{4^{k+1}} < \frac{1}{2}$ holds.

We need the computational space $O(kan)$ for finding the set $F \in \mathcal{F}$ that maximizes $\min\{|V_F|, |V_{\bar{F}}|\}$, and $O(s)$ for computing functions f_i. ◀

4 Satisfiability Algorithms for Syntactic Read-k-times BPs

4.1 Satisfiability Algorithm

In this section, we detail our satisfiability algorithm for syntactic read-k-times BPs and analyze its running time. We describe the outline of our algorithm. Our algorithm consists of two steps.

First, applying the decomposition algorithm in Lemma 2 with $a = 2k$, we decompose the input syntactic read-k-times BP B into a disjunction of at most m^{2k^2} BPs. Then, B is satisfiable iff at least one of these decomposed BPs is satisfiable. In addition, each decomposed BP consists of a conjunction of at most $2k^2$ BPs.

Second, we determine the satisfiability of each decomposed BP by checking whether there exists an assignment that satisfies all BPs. Let a decomposed BP be a conjunction of BPs

$\{B_1, \ldots, B_\ell\}$, where $\ell \leq 2k^2$. Applying Lemma 4 with $a = 2k$, we count the number of satisfiable assignments that satisfy all BPs.

Repeating the above operations for all decomposed BPs, we can determine the satisfiability of the input B.

▶ **Theorem 5** (Restatement of Theorem 1). *There exists a deterministic and polynomial space algorithm for a nondeterministic and syntactic read-k-times BP SAT with n variables and m edges that runs in time $O\left(\mathrm{poly}(n, m^{k^2}) \cdot 2^{(1-4^{-k-1})n}\right)$.*

Proof. Our algorithm consists of the following two steps: (1) decomposition and (2) satisfiability checking.

[Step 1: Decomposition]
Setting $a = 2k$ in Lemma 2, construct the set of BPs $\{B_{i,j}\}$ from the input B. Let f and $f_{i,j}$ be Boolean functions represented by B and $B_{i,j}$, respectively. Let $X_{i,j}$ be the set of variables that appear in $B_{i,j}$. Then, the following properties hold:

1. $f = \bigvee_{i \in [m^{2k^2}]} \bigwedge_{j \in [2k^2]} f_{i,j}$.
2. For each i and j, $|X_{i,j}|$ is at most $\lceil \frac{n}{2k} \rceil$. For each i, each variable x belongs to at most k sets of $\{X_{i,j}\}_{j=1,\ldots,2k^2}$.

The computational time required in Step 1 is at most $O\left(2k^2 m^{2k^2+1}\right)$.

[Step 2: Satisfiability Checking]
In order to check the satisfiability of B, we check whether there exists an assignment that satisfies all branching programs $B_{i,1}, \ldots, B_{i,2k^2}$ for each $i \in [m^{2k^2}]$. Let us consider a fixed i. We denote $B_{i,j}$, $f_{i,j}$, and $X_{i,j}$ simply by B_j, f_j, and X_j, respectively. Note that each function f_j can be computed in $O(m)$ time and $O(m)$ space by simulating the computation of B_j.

Our goal in this step is to determine whether there is an assignment that satisfies all f_j for $j \in [2k^2]$. By applying Lemma 4 and setting $a = 2k$, $t = O(m)$, and $s = O(m)$, we count the satisfiable assignments that satisfy all f_j in a time of at most

$$O\left(2^{2k^2} k^2 n\right) + O(k^2 m) \cdot 2^{(1-\frac{1}{4^{k+1}})n}.$$

Therefore, the running time of Step 2 is at most

$$m^{2k^2} \cdot \left\{O\left(2^{2k^2} k^2 n\right) + O(k^2 m) \cdot 2^{(1-\frac{1}{4^{k+1}})n}\right\} = \mathrm{poly}\left(n, m^{k^2}\right) \cdot 2^{(1-\frac{1}{4^{k+1}})n}.$$

Combining the analyses of Step 1 and Step 2, the running time of our algorithm is at most

$$O\left(2k^2 m^{2k^2+1}\right) + \mathrm{poly}\left(n, m^{k^2}\right) \cdot 2^{(1-\frac{1}{4^{k+1}})n} = \mathrm{poly}\left(n, m^{k^2}\right) \cdot 2^{(1-\frac{1}{4^{k+1}})n}.$$

Note that if a given B is a deterministic and syntactic read-k-times BP, then any satisfiable assignment of B satisfies only one conjunction part of the decomposed BPs. Then, the number of satisfiable assignments of B is equal to the sum of the results of Step 2. ◀

References

1 Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 375–388, 2016.

2 Vikraman Arvind and Rainer Schuler. The quantum query complexity of 0-1 knapsack and associated claw problems. In *Proceedings of the 14th International Symposium on Algorithms and Computation (ISAAC)*, pages 168–177, 2003.

3 David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989.

4 Paul Beame, Russell Impagliazzo, and Srikanth Srinivasan. Approximating AC^0 by small height decision trees and a deterministic algorithm for $\#\mathrm{AC}^0$ SAT. In *Proceedings of the 27th Conference on Computational Complexity (CCC)*, pages 117–125, 2012.

5 Paul Beame, T. S. Jayram, and Michael E. Saks. Time-space tradeoffs for branching programs. *J. Comput. Syst. Sci.*, 63(4):542–572, 2001.

6 Beate Bollig, Martin Sauerhoff, Detlef Sieling, and Ingo Wegener. Hierarchy theorems for $k$obdds and $k$ibdds. *Theoretical Computer Science*, 205(1-2):45–60, 1998.

7 Allan Borodin, Alexander A. Razborov, and Roman Smolensky. On lower bounds for read-k-times branching programs. *Computational Complexity*, 3:1–18, 1993.

8 Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A duality between clause width and clause density for SAT. In *Proceedings of the 21st Annual IEEE Conference on Computational Complexity (CCC)*, pages 252–260, 2006.

9 Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In *Revised Selected Papers from the 4th International Workshop on Parameterized and Exact Computation*, volume 5917 of *LNCS*, pages 75–85, 2009.

10 Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining circuit lower bound proofs for meta-algorithms. In *Proceedings of the 21st Annual IEEE Conference on Computational Complexity (CCC)*, pages 262–273, 2014.

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

12 Evgeny Dantsin, Edward A. Hirsch, and Alexander Wolpert. Algorithms for SAT based on search in Hamming balls. In *Proceedings of the 21st Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 141–151, 2004.

13 Evgeny Dantsin, Edward A. Hirsch, and Alexander Wolpert. Clause shortening combined with pruning yields a new upper bound for deterministic SAT algorithms. In *Proceedings of the 6th International Conference on Algorithms and Complexity (CIAC)*, pages 60–68, 2006.

14 Edward A. Hirsch. Exact algorithms for general CNF SAT. In *Encyclopedia of Algorithms*. 2008.

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

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

17 Stasys Jukna. A note on read-k times branching programs. *ITA*, 29(1):75–83, 1995.

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

19 Pavel Pudlák. Satisfiability - algorithms and logic. In *Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 129–141, 1998.

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

21 Martin Sauerhoff. Lower bounds for randomized read-k-times branching programs. In *Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 105–115, 1998.

22 Rainer Schuler. An algorithm for the satisfiability problem of formulas in conjunctive normal form. *Journal of Algorithms*, 54(1):40–44, 2005.

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

24 Avishay Tal. #SAT algorithms from shrinkage. *Electronic Colloquium on Computational Complexity (ECCC)*, 22(114), 2015.

25 Jayram S. Thathachar. On separating the read-k-times branching program hierarchy. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 653–662, 1998.

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

27 Ryan Williams. Nonuniform ACC circuit lower bounds. *Journal of ACM*, 61(1):2:1–2:32, 2014.