Theory of Computing Systems (2020) 64:1392–1407
https://doi.org/10.1007/s00224-020-09996-3

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

Atsuki Nagao[1] · Kazuhisa Seto[2] · Junichi Teruyama[3]

Published online: 27 July 2020

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. In a preliminary version of this paper, we provide a satisfiability algorithm for syntactic read-k-times branching programs with n variables and m edges that runs in time $O\left(\mathrm{poly}(n, m^{k^2}) \cdot 2^{(1-4^{-k-1})n}\right)$. In this paper, we improve the bounds for $k = 2$. More precisely, we show that the satisfiability of syntactic read-twice branching programs can be solved in time $O\left(\mathrm{poly}(n, m) \cdot 2^{5n/6}\right)$. Our algorithm is based on the decomposition technique shown by Borodin, Razborov and Smolensky [Computational Complexity, 1993].

Keywords Branching program · Read-k-times · Satisfiability · Moderately exponential time · Polynomial space

A preliminary version of this paper appeared in the Proceedings of the 28th International Symposium on Algorithms and Computation (ISAAC 2017).
This work was supported in part by JSPS KAKENHI (18K11170, 18K18003) and JST CREST Grant Number JPMJCR1402, Japan.

✉ Junichi Teruyama
junichi.teruyama@gmail.com

Atsuki Nagao
a-nagao@is.ocha.ac.jp

Kazuhisa Seto
seto@st.seikei.ac.jp

[1] Ochanomizu University, Tokyo, Japan

[2] Seikei University, Tokyo, Japan

[3] University of Hyogo, Kobe, Japan

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.

There exist two main restricted models, width-k and read-k-times BPs. A width-k BP is a leveled BP and each level has at most k nodes. Barrington [2] showed that any function in $\mathbf{NC^1}$ can be computed by width-5 BPs of polynomial length. Thus, it is a challenge to prove super-polynomial lower bounds for width-5 BPs. Despite of many efforts, such bounds are known for only width-2 BPs by Yao [20].

Read-k-times BPs have two models, *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. Beame, Saks, and Thathachar [3] showed that polynomial-size semantic read-twice BP can compute functions requiring exponential size on any syntactic read-k-times BP. Thus, the semantic model is substantially stronger than the syntactic model. For the semantic case, Jukna [12] proved an exponential lower bound for syntactic read-k-times BPs over large domain D, where $|D| = 2^{3k+10}$. In case of read-once BPs, Cook, Edmonds, Medabalimi, and Pitassi [7] proved exponential lower bounds for $|D| = 3$.

The syntactic model is well studied. Borodin, Razborov, and Smolensky [5] exhibited an explicit function of the lower bound of $\exp\left(\Omega\left(\frac{n}{k^3 4^k}\right)\right)$. Jukna [11] 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 [18] 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 [16] proved the exponential lower bound for randomized read-k-times BPs with a two-sided error.

The satisfiability of BPs (BP SAT), given a BP, is to determine 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 $\mathcal{C}$ be a class of a circuit. Given a circuit in $\mathcal{C}$, $\mathcal{C}$-SAT is the determination of whether there exists an assignment to the input variables such that the circuit outputs 1. Williams [19] showed that to obtain $\mathbf{NEXP} \not\subseteq \mathcal{C}$, it suffices to develop an $O\left(2^{n-\omega(\log n)}\right)$ time algorithm for $\mathcal{C}$-SAT. By combining Barrignton's theorem [2], if we would like to prove $\mathbf{NEXP} \not\subseteq \mathbf{NC^1}$, it 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 design of a fast algorithm for BP SAT is one of the important tasks in the field of computational complexity.

We have another motivation for designing satisfiability algorithms. A technique for proving a lower bound against a class of circuit $\mathcal{C}$ is deeply connected to the design of an algorithm for $\mathcal{C}$-SAT. For example, Paturi, Pudlák, and Zane [14] proposed the satisfiability coding lemma, and applied it to design a satisfiability algorithm for CNF-SAT and to prove a lower bound for the parity function on depth-3 circuits. For $\mathbf{AC}^0$, Håstad's switching lemma [8] helps us to prove an exponential lower bounds on the parity function. Impagliazzo, Matthews, and Paturi [10], by extending it, presented a satisfiability algorithm for $\mathbf{AC}^0$ and improved the correlation lower bound of $\mathbf{AC}^0$ with parity. Santhanam [15] presented a moderately exponential time algorithm for De Morgan formula SAT based on lower bound techniques by Subbotovskaya [17] and Håstad [9]. According to these results, we believe that any proof technique for a lower bound is translated into designing a satisfiability algorithm.

In this paper, we consider the satisfiability of syntactic read-k-times BPs. There exists no satisfiability algorithm for $k \geq 2$ faster than a brute-force search, while the technique of lower bound by Borodin, Razborov, and Smolensky [5] is known. We present a moderately exponential time satisfiability algorithm based on the lower bound technique for read-k-times BPs where $k \geq 2$. As a result, we get the following theorem.

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

When $k = 2$, we improve the running time of our algorithm.

Theorem 2 *There exists a deterministic and polynomial space algorithm for a nondeterministic and syntactic read-twice BP SAT with n variables and m edges that runs in time* $O\left(\mathrm{poly}(n, m) \cdot 2^{5n/6}\right)$.

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 [5]. 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 $\mathcal{F} = \{f_1, f_2, \ldots, f_\ell\}$, where $\ell \leq 2k^2$. **[Step 2: Satisfiability Checking]** To check the satisfiability of $\mathcal{F}$, we find if there is 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

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 [4] 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 $k \geq 2$ [4]. Nagao, Seto, and Teruyama [13] 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 [6] showed that general BP SAT with $o(n^2)$ nodes can be determined in time $O\left(2^{n-\omega(\log n)}\right)$.

1.3 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 [5]. 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 and improve this algorithm for $k = 2$ by modifying the satisfiability algorithm of Section 3.

2 Preliminaries

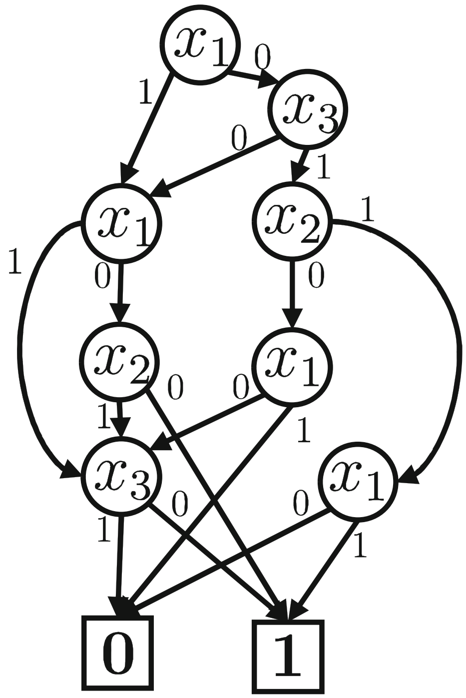
For a positive integer n, we denote by $[n]$ a set of integers $\{1, 2, \ldots, n\}$. For a set S, $|S|$ denotes the cardinality of S. Let $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables, and for $x \in X$, $\overline{x}$ denotes the negation of x. A *branching program* (BP), denoted by $B = (V, E)$, 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 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 node except the two sink nodes in V has exactly two outgoing edges, a 0-edge and 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.

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 subgraph of B that contains v, w, and every nodes and edges contained in all v-w paths in B. Given a BP B and nodes $v, w \in V$, we can construct $\langle B, v, w\rangle$ in $O(|B|)$ time as follows:

1. Find the subset V' of V such that V' contains every node that is reachable from v and to w by the breadth-first search.
2. Obtain the subgraph of B induced by V'.

Fig. 1 Syntactic read-twice branching program



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', $\alpha \circ \alpha'$ denotes the *composition* of α and α' defined as follows: $\alpha \circ \alpha'(i) = \alpha'(i)$ if $x_i \in S(\alpha')$, and $\alpha \circ \alpha'(i) = \alpha(i)$ otherwise. For instance, when $\alpha = (1, *, 1)$ and $\alpha' = (*, *, 0)$, we have $\alpha \circ \alpha' = (1, *, 0)$.

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 [5]. 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 $\lceil n/a \rceil$ 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 [5]. We will use this algorithm as a module to solve the syntactic read-k-times BP SAT in Section 4.1.

Lemma 1 (Theorem 1 in [5]) *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 $\{X_{i,j}\}_{j=1,\ldots,ka}$.*

We revisit the algorithm given in the proof of Theorem 1 in [5]. 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 find $X(v, w)$ by dynamic programming for each pair of nodes $v, w \in V$.

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

(a) For each j with $1 \leq j \leq \ell + 1$, we have $|X(v_j, w_j)| < n/a$.

(b) For each j with $1 \leq j \leq \ell$, we have $|X(v_j, v_{j+1})| \geq n/a$,

where we set v_1 as the root and $w_{\ell+1}$ as the 1-sink. For each trace, any variable x belongs to at most k sets $X(v_j, w_j)$, because B is a syntactic read-k-times BP. Note that any path from r to the 1-sink contains a unique trace. By property (b), for each trace, we have

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

where $w_{\ell+1}$ is the 1-sink. Since each variable belongs to at most k sets, the left-hand side can be bounded above by kn. This leads that the length ℓ of traces is at most ka.

Let $\mathcal{T}$ be the set of all traces. We construct $\mathcal{T}$ as follows. Enumerate all sequences of edges in E with the length of at most ka. For each sequence, check whether it satisfies the conditions (a) and (b) by using $X(v, w)$. If it satisfies the conditions, then put it in $\mathcal{T}$. Note that the number of sequences needed to check is at most $\sum_{\ell=1}^{ka} \binom{m}{\ell} \leq m^{ka}$. Thus, we have $|\mathcal{T}| \leq m^{ka}$.

For each trace $T = (e_1 = (w_1, v_2), \ldots, e_\ell = (w_\ell, v_{\ell+1})) \in \mathcal{T}$ and $1 \leq j \leq \ell$, 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 $B_{T,j}$. Thus, 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. Recalling that the left-hand side of inequality (1) is bounded above by kn and we have $\ell \leq ka$, $\ell = ka$ 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.

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

Proof Let us analyze the running time of the above construction. First, for each pair of nodes $v, w \in V$, we find $X(v, w)$ by dynamic programming in $O(m)$ time. Therefore, the running time for enumerating all $X(v, w)$ is $O(m^3)$. The running time for construction of $\mathcal{T}$ is $O(kam^{ka})$, because we have m^{ka} sequences and check the conditions for each sequence in $O(ka)$ time. For each trace $T \in \mathcal{T}$, we construct at most ka branching programs $B_{t,j}$ in $O(kam)$ time. Since we have $|\mathcal{T}| \leq m^{ka}$, we construct at most kam^{ka} branching programs in $O(kam^{ka+1})$ time. Totally, the total running time is at most $O(m^3) + O(kam^{ka}) + O(kam^{ka+1}) = O(kam^{ka+1})$. $\square$

Next, we give the satisfiability algorithm for a specific class Boolean functions. Let a and k be positive integers with $a \le n$. We are given a set of ka functions $\{f_i\}$ ($i \in [ka]$) with 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.

Our task is to count the satisfiable assignments of the function $f = \bigwedge_{i \in [ka]} f_i$.

Algorithm 1

Require: ka Boolean functions $\{f_i\}$ ($i \in [ka]$) with the properties of the statement in Lemma 3.
Ensure: The number of assignments that satisfy a function $f = \bigwedge f_i$.
1: $N := 0$.
2: Find a set $F \subseteq [ka]$ that maximizes $\min\{|V_F|, |V_{\bar{F}}|\}$ by exhaustive search, where V_F is $\left(\bigcup_{i \in F} X_i\right) \setminus \left(\bigcup_{i \in \bar{F}} X_i\right)$.
3: Let $Y := X \setminus (V_F \cup V_{\bar{F}})$.
4: **for** $\forall \alpha \in \{0, 1, *\}^n$ such that $S(\alpha) = Y$ **do**
5: $\quad A_F := \emptyset$, $A_{\bar{F}} := \emptyset$.
6: $\quad$**for** $\forall \alpha_F \in \{0, 1, *\}^n$ such that $S(\alpha_F) = V_F$ **do**
7: $\quad\quad$**if** $f_i|_\alpha(\alpha_F) = 1$ for all $i \in F$ **then**
8: $\quad\quad\quad$Insert α_F into A_F.
9: $\quad\quad$**end if**
10: $\quad$**end for**
11: $\quad$**for** $\forall \alpha_{\bar{F}} \in \{0, 1, *\}^n$ such that $S(\alpha_{\bar{F}}) = V_{\bar{F}}$ **do**
12: $\quad\quad$**if** $f_i|_\alpha(\alpha_{\bar{F}}) = 1$ for all $i \in \bar{F}$ **then**
13: $\quad\quad\quad$Insert $\alpha_{\bar{F}}$ into $A_{\bar{F}}$.
14: $\quad\quad$**end if**
15: $\quad$**end for**
16: $\quad N := N + |A_F| \cdot |A_{\bar{F}}|$
17: **end for**
18: **return** N

We describe a satisfiability algorithm for function f. See Algorithm 1. To explain our algorithm, we need some notations. Let us consider a subset F of $[ka]$. Note that the number of such sets is 2^{ka}. For $F \subseteq [ka]$, $\bar{F}$ 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 the definition, for any $F \in \mathcal{F}$, V_F and $V_{\bar{F}}$ are disjoint.

First, our algorithm finds a subset F of $[ka]$ that maximizes $\min\{|V_F|, |V_{\bar{F}}|\}$ at Line 2. For this F, let Y be a set of variables $X \setminus \left(V_F \cup V_{\bar{F}}\right)$. That is, any variable $x \in Y$ appears both in X_i and X_j for some $i \in F$ and $j \in \bar{F}$. Let us pick up a partial assignment α whose support is Y. All functions $f_i|_\alpha$ 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 $f_i|_\alpha(\alpha_F) = 1$ for all $i \in F$. We find A_F by an exhaustive search, i.e., checking the value $f_i|_\alpha(\alpha_F)$ for all $i \in F$ and all partial assignments with $S(\alpha_F) = V_F$ in Lines 6–10. Similarly, we find $A_{\bar{F}}$ which is a set of partial assignments $\alpha_{\bar{F}}$ such that $S(\alpha_{\bar{F}}) = V_{\bar{F}}$ and $f_i|_\alpha(\alpha_{\bar{F}}) = 1$ for all $i \in \bar{F}$ by an exhaustive search in Lines 11–15. For any $\alpha_F \in A_F$ and any $\alpha_{\bar{F}} \in A_{\bar{F}}$, we have $f(\alpha \circ \alpha_F \circ \alpha_{\bar{F}}) = 1$ holds, since $f = \bigwedge_{i=1}^{ka} f_i$. Therefore, the number of assignments that satisfy f and contain a partial assignment α is exactly $|A_F| \cdot |A_{\bar{F}}|$. In the for-loop (Lines 4–17), we sum up $|A_F| \cdot |A_{\bar{F}}|$ for all partial assignments with support Y and then obtain the number of all assignments that satisfy f.

We give the other key lemma.

Lemma 3 *Let a and k be positive integers with $a \leq n$. Suppose that we are given a set of ka functions $\{f_i\}$ ($i \in [ka]$) 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., $|X_i| \leq \lceil n/a \rceil$.*
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 Let us 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 $\sum_i |X_i| \leq k(n-1)$. This implies that there exists a set X_i such that $|X_i| \leq (n-1)/a < \lceil n/a \rceil$. Then, we can put the variable x into the set X_i while preserving the properties.

First, Algorithm 1 requires the computational space $O(kan)$ at Line 2, and $O(s)$ for computing functions $f_i|_\alpha(\alpha_F)$ and $f_i|_\alpha(\alpha_{\bar{F}})$ at Lines 7 and 12.

Next, we give the analysis of the complexity of Algorithm 1. At Line 2, we compute a subset $F \in [ka]$ that maximizes $\min\{|V_F|, |V_{\bar{F}}|\}$. For each $F \subseteq [ka]$, we have $\min\{|V_F|, |V_{\bar{F}}|\}$ in $O(kan)$ time. Therefore, Line 2 requires $O(2^{ka}kan)$ time since the number of subsets of $[ka]$ is 2^{ka}.

Let us fix a partial assignment α with support Y in the for-loop of Lines 5–17. We construct A_F by checking the value $f_i|_\alpha(\alpha_F)$ for all $i \in F$ and all partial assignments with $S(\alpha_F) = V_F$ in Lines 6–10. Thus, Lines 6–10 require $t \cdot |F| \cdot 2^{|V_F|} \leq tka \cdot 2^{|V_F|}$ time. Similarly, Lines 11–15 require at most $tka \cdot 2^{|V_{\bar{F}}|}$ time to construct $A_{\bar{F}}$. Since we have $2^{|Y|}$ partial assignments with support Y, Lines 4–17 require at most $tka \cdot 2^{|Y|} \cdot \left(2^{|V_F|} + 2^{|V_{\bar{F}}|}\right)$ time. Because $|Y| + |V_F| + |V_{\bar{F}}| = n$, we have

$$2^{|Y|} \cdot \left(2^{|V_F|} + 2^{|V_{\bar{F}}|}\right) \leq 2^{|Y|} \cdot \left(2^{\max\{|V_F|,|V_{\bar{F}}|\}+1}\right) = 2^{n-\min\{|V_F|,|V_{\bar{F}}|\}+1}.$$

Totally, the running time of Algorithm 1 is at most

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

We will show that $\max_{F\subseteq[ka]}\min\{|V_F|,|V_{\bar{F}}|\}$ is at least $\frac{2}{4^{k+1}}\left(1-\frac{k}{a}\right)n-\frac{1}{2}$. It follows that the second term of the running time is

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

Thus, it leads the lemma.

To analyze $\max_{F\subseteq[ka]}\min\{|V_F|,|V_{\bar{F}}|\}$, let us suppose that n is even for simplicity. (In the case when n is odd, we also obtain the same result in a similar way.) Let S be the set of variables $\{x_1,\ldots,x_{n/2}\}$ and L be the set of variables $\{x_{(n/2)+1},\ldots,x_n\}$. Now, we define good/bad pairs of variables. This notation is used in the proof of Theorem 6 in [5]. A pair $(x,x')\in S\times L$ is *good* iff there is no $i\in[ka]$ such that X_i contains both x and x', and *bad* otherwise. For each i, we have $|S\cap X_i|\cdot|L\cap X_i|$ pairs $(x,x')\in S\times L$ such that X_i contains both x and x'. Since $|S\cap X_i|+|L\cap X_i|=|X_i|\le\left\lceil\frac{n}{a}\right\rceil$ hold, we have

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

By the union bound, the number of bad pairs is at most $\frac{ka}{4}\cdot\left\lceil\frac{n}{a}\right\rceil^2$. Thus, using $\left\lceil\frac{n}{a}\right\rceil<\frac{n}{a}+1$ and $a\le n$, the number of good pairs is at least

$$|S|\cdot|L|-(\#\text{bad pairs})\ge\frac{n^2}{4}-\frac{ka}{4}\cdot\left\lceil\frac{n}{a}\right\rceil^2>\frac{1}{4}\left(1-\frac{k}{a}\right)n^2-\frac{3k}{4}n.$$

Let us consider that a subset $F\subseteq[ka]$ is chosen uniformly at random. For any variable $x\in X$, we have

$$\mathbf{Pr}[x\in V_F]=\prod_{i\in\{i|x\in X_i\}}\mathbf{Pr}[i\in F]\ge 2^{-k},$$

since x belongs to at most k sets X_i. Also, we have $\mathbf{Pr}[x\in V_{\bar{F}}]\ge 2^{-k}$. Therefore, for each good pair $(x,x')\in S\times L$, we have $\mathbf{Pr}[x\in V_F,x'\in V_{\bar{F}}]\ge 4^{-k}$. Hence, the average number of good pairs $(x,x')\in S\times L$ such that $x\in V_F,x'\in V_{\bar{F}}$ is at least

$$\frac{1}{4^k}\left[\frac{1}{4}\left(1-\frac{k}{a}\right)n^2-\frac{3kn}{4}\right].$$

This implies that there exists a set $F\subseteq[ka]$ such that

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

For such a set F, we have

$$\begin{aligned} \min\{|S\cap V_F|,|L\cap V_{\bar{F}}|\} &= \frac{|S\cap V_F|\cdot|L\cap V_{\bar{F}}|}{\max\{|S\cap V_F|,|L\cap V_{\bar{F}}|\}} \\ &\ge\frac{2}{4^k n}\left[\frac{1}{4}\left(1-\frac{k}{a}\right)n^2-\frac{3kn}{4}\right], \end{aligned}$$

since $\max\{|S \cap V_F|, |L \cap V_{\bar{F}}|\} \le \max\{|S|, |L|\} = \frac{n}{2}$. Thus, we have

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

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

$$\max_{F \subseteq [ka]} \min\{|V_F|, |V_{\bar{F}}|\} \ge \frac{2}{4^{k+1}}\left(1-\frac{k}{a}\right)n - \frac{1}{2}$$

and the proof is complete. □

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 1 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 $\{B_1, \dots, B_\ell\}$, where $\ell \le 2k^2$. Applying Lemma 3 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 3 (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 1, 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 $\{X_{i,j}\}_{j=1,\dots,2k^2}$.

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 $B_{i,1}, \dots, B_{i,2k^2}$ 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 3 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

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

Therefore, the running time of Step 2 is at most

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

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

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

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. □

4.2 Improved Algorithm for $k = 2$

By Theorem 1, we can solve the satisfiability of syntactic read-twice BPs in time $\text{poly}(n, m) \cdot 2^{63n/64}$. We improve this bounds to $\text{poly}(n, m) \cdot 2^{5n/6}$.

Theorem 4 (Restatement of Theorem 2) *There exists a deterministic and polynomial space algorithm for a nondeterministic and syntactic read-twice BP SAT with n variables and m edges that runs in time* $O\left(\text{poly}(n, m) \cdot 2^{5n/6}\right)$.

To prove theorem, we improve the analysis of the running time of Algorithm 1 when $k = a = 2$. Note that Lemma 3 does not give the upper bound of the running time when $k = a$ holds.

Lemma 4 *Suppose that we are given a set of four functions f_i ($i = 1, 2, 3, 4$) that satisfy the following properties:*

1. *Each f_i depends on only at most $\lceil n/2 \rceil$ variables $X_i \subset X$, i.e., $|X_i| \le \lceil n/2 \rceil$.*
2. *Each variable x belongs to at most two 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 $O(n) + O(t) \cdot 2^{5n/6}$ and space $O(s + n)$.

Proof By the same way of the proof of Lemma 3, we assume that each variable x belongs to at least one set X_i. By Lemma 3, Algorithm 1 requires the computational space $O(s + kan) = O(s + n)$. Recall that for a set $F \subseteq [4]$, V_F is a set of variables that belong to only $\bigcup_{i \in F} X_i$. The running time of Algorithm 1 is at most

$$O(n) + 8t \cdot 2^{n - \min\{|V_F|, |V_{\bar{F}}|\}}$$

from the proof of Lemma 3 with $k = a = 2$. To complete the proof, we need to show that

$$\max_{F \subseteq [4]} \min\{|V_F|, |V_{\bar{F}}|\} \ge \frac{n}{6}.$$

We show the following lemma and give the proof later.

Lemma 5 *Let w_i and $v_{i,j}$ be non-negative real values, where $i, j \in [4]$ and $i \neq j$ such that the following equations hold.*

1. $v_{i,j} = v_{j,i}$,
2. *for each i, $w_i + \sum_{j \neq i} v_{i,j} \le 1/2$,*
3. $\sum_{i \in [4]} w_i + \sum_{i,j \in [4], i<j} v_{i,j} = 1$.

Let $x_{i,j} = w_i + w_j + v_{i,j}$. Then,

$$\max\left\{\min\{x_{1,2}, x_{3,4}\}, \min\{x_{1,3}, x_{2,4}\}, \min\{x_{1,4}, x_{2,3}\}\right\} \ge 1/6$$

holds.

For $i \in [4]$, we set w_i as the proportion of variables in only X_i. For $i, j \in [4]$, we set $v_{i,j}$ as the proportion of variables in both X_i and X_j. Under this setting, $x_{i,j} = w_i + w_j + v_{i,j}$ corresponds that the proportion of variables in only X_i or X_j. Thus, we have $x_{i,j} = \frac{|V_F|}{n}$ with $F = \{i, j\}$. In order to apply Lemma 5, we need to confirm all conditions hold under this setting. By the symmetry of the setting of $v_{i,j}$, condition 1 clearly holds. The left side term $w_i + \sum_{j \neq i} v_{i,j}$ of condition 2 corresponds to a portion of variables that contains in X_i. By the property 1 of the statement of Lemma, condition 2 holds. The left side of condition 3 corresponds to a portion of variables that is in some set X_i, then condition 3 clearly holds. Thus, Lemma 5 implies that there exits a subset F of size two such that

$$\min\{|V_F|, |V_{\bar{F}}|\} \geq n/6.$$

It completes the proof. □

Proof of Lemma 5 Without loss of generality, we can assume that one of the following two cases holds: (i) $\min\{x_{1,2}, x_{3,4}\} = x_{1,2}$, $\min\{x_{1,3}, x_{2,4}\} = x_{1,3}$, $\min\{x_{1,4}, x_{2,3}\} = x_{1,4}$ and (ii) $\min\{x_{1,2}, x_{3,4}\} = x_{1,2}$, $\min\{x_{1,3}, x_{2,4}\} = x_{1,3}$, $\min\{x_{1,4}, x_{2,3}\} = x_{2,3}$.

(i) Let us assume that $\min\{x_{1,2}, x_{3,4}\} = x_{1,2}$, $\min\{x_{1,3}, x_{2,4}\} = x_{1,3}$, $\min\{x_{1,4}, x_{2,3}\} = x_{1,4}$. By $x_{1,i} = w_1 + w_i + v_{1,i}$ for each $i \in \{2, 3, 4\}$, we have $\max\{x_{1,2}, x_{1,3}, x_{1,4}\} = w_1 + \max_{i \in \{2,3,4\}}\{w_i + v_{1,i}\}$. Recall the condition 2 that $w_i + \sum_{j \neq i} v_{i,j} \leq 1/2$ holds for each i. Summing up this inequality for $i = 2, 3, 4$, we have

$$\sum_{i \in \{2,3,4\}} \left(w_i + v_{1,i}\right) + 2 \cdot (v_{2,3} + v_{2,4} + v_{3,4}) \leq \frac{3}{2}$$

$$v_{2,3} + v_{2,4} + v_{3,4} \leq \frac{3}{4} - \frac{\sum_{i \in \{2,3,4\}} \left(w_i + v_{1,i}\right)}{2}. \quad (2)$$

Recalling we have condition 3 that $\sum_{i \in [4]} w_i + \sum_{i,j \in [4], i<j} v_{i,j} = 1$, we have

$$\begin{aligned} 1 &= \sum_{i \in [4]} w_i + \sum_{i,j \in [4], i<j} v_{i,j} \\ &= w_1 + \sum_{i \in \{2,3,4\}} \left(w_i + v_{1,i}\right) + v_{2,3} + v_{2,4} + v_{3,4} \\ &\leq w_1 + \frac{\sum_{i \in \{2,3,4\}} \left(w_i + v_{1,i}\right)}{2} + \frac{3}{4}. \end{aligned}$$

The last inequality led by inequality (2). Simplifying the above inequality, we have

$$\sum_{i \in \{2,3,4\}} \left(w_i + v_{1,i}\right) \geq \frac{1}{2} - 2w_1. \quad (3)$$

Therefore, we have

$$\begin{aligned} w_1 + \max_{i \in \{2,3,4\}}\{w_i + v_{1,i}\} &\geq w_1 + \frac{1}{3} \sum_{i \in \{2,3,4\}} \left(w_i + v_{1,i}\right) \\ &\geq w_1 + \frac{1}{6} - \frac{2}{3} w_1 \quad (\because \text{By inequality (3)}) \\ &= \frac{1}{6} + \frac{1}{3} w_1 \geq \frac{1}{6}. \end{aligned}$$

(ii) Let we assume that $\min\{x_{1,2}, x_{3,4}\} = x_{1,2}$, $\min\{x_{1,3}, x_{2,4}\} = x_{1,3}$ and $\min\{x_{1,4}, x_{2,3}\} = x_{2,3}$ hold. Recalling condition 3 again, we have

$$\begin{aligned} 1 &= \sum_{i\in[4]} w_i + \sum_{i,j\in[4],i<j} v_{i,j} \\ &= x_{1,2} + x_{1,3} + x_{2,3} - \sum_{i\in\{1,2,3\}} w_i + w_4 + \sum_{i\in\{1,2,3\}} v_{i,4} \\ &\leq x_{1,2} + x_{1,3} + x_{2,3} + 1/2. \qquad (\because \text{By condition 2}) \end{aligned}$$

Thus, we have $x_{1,2} + x_{1,3} + x_{2,3} \geq 1/2$ and this implies that $\max\{x_{1,2}, x_{1,3}, x_{2,3}\} \geq 1/6$ holds. □

Proof of Theorem 2 Improved algorithm for $k = 2$ also consists of the following two steps: (1) decomposition and (2) satisfiability checking.

[Step 1: Decomposition]

Setting $a = 2$ in Lemma 1, construct a set of BPs $\{B_{i,j} \mid i \in [m^4], j \in [4]\}$ from the input BP 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 $B_{i,j}$. By Lemma 1, the following properties hold:

1. $f = \bigvee_{i\in[m^4]} \bigwedge_{j\in[4]} f_{i,j}$.
2. For each i and j, $|X_{i,j}|$ is at most $n/2$.
3. For each i, each variable x belongs to at most two sets of $\{X_{i,1}, X_{i,2}, X_{i,3}, X_{i,4}\}$.

The running time required in Step 1 is at most $\mathrm{poly}(m)$.

[Step 2: Satisfiability Checking]

In order to check the satisfiability of B, we check whether there exists an assignment that satisfies all four branching programs $B_{i,1}$, $B_{i,2}$, $B_{i,3}$ and $B_{i,4}$ for each $i \in [m^4]$. 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.

Now, our task is to decide whether there is an assignment that satisfies all f_j, where $j \in [4]$. By applying Lemma 4 and setting $t = O(m)$, and $s = O(m)$, we count the satisfiable assignments that satisfy all f_j in a time of at most

$$O(n) + O(m) \cdot 2^{5n/6}.$$

Therefore, the running time of Step 2 is at most

$$m^4 \cdot \left\{O(n) + O(m) \cdot 2^{5n/6}\right\} = \mathrm{poly}(n, m) \cdot 2^{5n/6}.$$

Since the running time of Step 1 is dominated by the one of Step 2, the total of running time for solving the satisfiability problem for read-twice BPs is at most $\mathrm{poly}(n, m) \cdot 2^{5n/6}$. □

References

1. Abboud, A., Hansen, T.D., Williams, V.V., Williams, R.: 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, pp. 375–388 (2016)
2. Barrington, D.A.M.: Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. J. Comput. Syst. Sci. **38**(1), 150–164 (1989)
3. Beame, P., Jayram, T.S., Saks, M.E.: Time-space tradeoffs for branching programs. J. Comput. Syst. Sci. **63**(4), 542–572 (2001)
4. Bollig, B., Sauerhoff, M., Sieling, D., Wegener, I.: Hierarchy theorems for kobdds and kibdds. Theor. Comput. Sci. **205**(1-2), 45–60 (1998)
5. Borodin, A., Razborov, A.A., Smolensky, R.: On lower bounds for read-k-times branching programs. Comput. Complex. **3**, 1–18 (1993)
6. Chen, R., Kabanets, V., Kolokolova, A., Shaltiel, R., Zuckerman, D.: Mining circuit lower bound proofs for meta-algorithms. In: Proceedings of the 21st Annual IEEE Conference on Computational Complexity (CCC), pp. 262–273 (2014)
7. Cook, S., Edmonds, J., Medabalimi, V., Pitassi, T.: Lower bounds for nondeterministic semantic Read-Once branching programs. In: Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP), Leibniz International Proceedings in Informatics (LIPIcs), vol. 55, pp. 36:1–36:13 (2016)
8. Håstad, J.: Almost optimal lower bounds for small depth circuits. In: Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA, pp. 6–20 (1986)
9. Håstad, J.: The shrinkage exponent of de morgan formulas is 2. SIAM J. Comput. **27**(1), 48–64 (1998)
10. Impagliazzo, R., Matthews, W., Paturi, R.: A satisfiability algorithm for AC^0 . In: Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 961–972 (2012)
11. Jukna, S.: A note on read-k times branching programs. ITA **29**(1), 75–83 (1995)
12. Jukna, S.: A nondeterministic space-time tradeoff for linear codes. Inf. Process. Lett. **109**(5), 286–289 (2009)
13. Nagao, A., Seto, K., Teruyama, J.: A moderately exponential time algorithm for k-IBDD satisfiability. In: Proceedings of Algorithms and Data Structures - 14th International Symposium (WADS), pp. 554–565 (2015)
14. Paturi, R., Pudlák, P., Zane, F.: Satisfiability coding lemma. Chicago Journal Theoretical Computer Science 1999 (1999)
15. Santhanam, R.: 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), pp. 183–192 (2010)
16. Sauerhoff, M.: Lower bounds for randomized read-k-times branching programs. In: Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science (STACS), pp. 105–115 (1998)
17. Subbotovskaya, B.A.: Realizations of linear functions by formulas using and, or, not. Soviet Mathematics Doklady **2**, 110–112 (1961)
18. Thathachar, J.S.: On separating the read-k-times branching program hierarchy. In: Proceedings of the 30th Annual ACM Symposium on the Theory of Computing (STOC), pp. 653–662 (1998)
19. Williams, R.: Improving exhaustive search implies superpolynomial lower bounds. SIAM J. Comput. **42**(3), 1218–1244 (2013)
20. Yao, A.C.: Lower bounds by probabilistic arguments (extended abstract). In: Proceedings of the 24th Annual Symposium on Foundations of Computer Science (STOC), pp. 420–428 (1983)