Electronic Colloquium on Computational Complexity, Report No. 178 (2019)

Almost Tight Lower Bounds on Regular Resolution Refutations of Tseitin Formulas for All Constant-Degree Graphs

Dmitry Itsykson* dmitrits@pdmi.ras.ru

Artur Riazanov* aariazanov@gmail.com

Danil Sagunov* danilka.pro@gmail.com

Petr Smirnov† comradepetr@gmail.com

December 4, 2019

Abstract

We show that the size of any regular resolution refutation of Tseitin formula $\mathrm{T}(G, c)$ based on a graph G is at least $2^{\Omega(\mathrm{tw}(G)/\log n)}$, where n is the number of vertices in G and $\mathrm{tw}(G)$ is the treewidth of G. For constant degree graphs there is known upper bound $2^{\mathcal{O}(\mathrm{tw}(G))}$ [1, 13], so our lower bound is tight up to a logarithmic factor in the exponent.

In order to prove this result we show that any regular resolution proof of Tseitin formula $\mathrm{T}(G, c)$ of size S can be converted to a read-once branching program computing satisfiable Tseitin formula $\mathrm{T}(G, c')$ of size $S^{\mathcal{O}(\log n)}$. Then we show that any read-once branching program computing satisfiable Tseitin formula $\mathrm{T}(G, c')$ has size at least $2^{\Omega(\mathrm{tw}(G))}$; the latter improves the recent result of Glinskih and Itsykson [15].

1 Introduction

In this paper we study Tseitin formulas encoding in CNF the following parity principle: any graph has an even number of vertices with an odd degree. Tseitin formula is based on an undirected graph $G(V, E)$ and a charge function $c\colon V \to \{0, 1\}$, the variables of $\mathrm{T}(G, c)$ correspond to the edges of the graph. The formula itself is the conjunction of the parity conditions of the vertices of G stating that the sum of the variables of the edges incident to v equals $c(v)$ modulo 2. We assume (and this is quite usual assumption) that degrees of all vertices of G do not exceed a constant. In that case the Tseitin formula $\mathrm{T}(G, c)$ has $\mathcal{O}(|V|)$ clauses and $\mathcal{O}(|V|)$ variables. There is a simple criterion for the satisfiability: a Tseitin formula $\mathrm{T}(G, c)$ is satisfiable iff for every connected component of G the sum of the values of $c(v)$ is even [28].

Unsatisfiable Tseitin formulas are widely studied in proof complexity. For specific families of graphs Tseitin formulas require exponentially long proofs in many proof systems [28, 4, 24, 18, 9, 16]. In this paper we consider resolution proof system and two its subsystems: regular resolution and tree-like resolution. For unsatisfiable CNF formula φ we denote by $S(\varphi)$, $S_R(\varphi)$ and $S_T(\varphi)$ the minimal size of unrestricted, regular and tree-like resolution proof of φ respectively. The following

*Saint Petersburg Department of Steklov Institute of Mathematics of Russian Academy of Sciences

†Saint Petersburg State University

ISSN 1433-8092

inequalities trivially hold: $S_T(\varphi) \geq S_R(\varphi) \geq S(\varphi)$. Let $w(\varphi)$ denote the minimal resolution width of φ.

Galesi, Toran and Talebanfard [13] study the characterizations of resolution width, variable space and depth of Tseitin formulas in terms of cop-robber games on the underlying graph. The results of [13] imply that for constant degree graphs the resolution width of $\mathrm{T}(G, c)$ equals the treewidth of G up to a constant factor:

$$w(\mathrm{T}(G, c)) = \Theta(\mathrm{tw}(G)).$$

In this paper we are interested in the shortest proof size of Tseitin formulas and our goal is to determine its dependence from properties of graphs. For tree-like proofs the size-width relation by Ben-Sasson and Wigderson [5] implies the lower bound $S_T(\mathrm{T}(G, c)) \geq 2^{\Omega(\mathrm{tw}(G))}$. There is also known upper bound $S_T(\mathrm{T}(G, c)) \leq 2^{\mathcal{O}(\mathrm{tw}(G) \log |V|)}$ [3, 21]; notice that the upper and the lower bounds do not match.

Alekhnovich and Razborov [1] proved that $S_R(\mathrm{T}(G, c)) \leq 2^{\mathcal{O}(\mathrm{tw}(G))}\mathrm{poly}(|V|)$ and, moreover, they showed that a regular resolution proof of $\mathrm{T}(G, c)$ can be generated in $2^{\mathcal{O}(\mathrm{tw}(G))}\mathrm{poly}(|V|)$ steps. The size-width relation [5] implies that the lower bound $S(\mathrm{T}(G, c)) \geq 2^{\Omega(\mathrm{tw}(G))}$ holds for graphs with large treewidth $\mathrm{tw}(G) = \Omega(|V|)$. On the one hand, random constant-degree graphs are expanders with high probability, hence with high probability they have $\mathrm{tw}(G) = \Omega(|V|)$. On the other hand, this approach, for example, does not yield lower bounds for $n \times n$ grid graphs. In 2001 Dantchev and Riis [11] proved that $S(\mathrm{T}(\mathrm{Grid}_{n\times n}, c)) = 2^{\Omega(n)}$. Alekhnovich and Razborov [1] proved the inequality $S(\mathrm{T}(G, c)) \geq 2^{\Omega(\mathrm{tw}(G))}$ for graphs that can be covered by cycles of constant length such that every edge is covered at most constant number of times (notice that $\mathrm{Grid}_{n\times n}$ has this property). However, for other graphs such lower bounds are not known even for regular resolution.

There is known approach that allows to estimate the resolution complexity of Tseitin formulas using treewidth of the underlying graph and the improved Grid Minor Theorem. The latter states that every graph G has a $t \times t$ grid as a minor, where $t = \Omega(\mathrm{tw}(G)^\delta)$ and δ is a constant; the latest improvement [10] establishes the theorem for $\delta = 1/10$, however, it is known that δ can not be greater than $1/2$. Håstad [18] proved that any d-depth Frege refutation of Tseitin formulas based on an $n \times n$ grid graph has size $2^{n^{\Omega(1/d)}}$ for $d \leq \frac{c \log n}{\log \log n}$, where c is a constant. Galesi et al. [12] have recently shown that any d-depth Frege refutation of a Tseitin formula $\mathrm{T}(G, c)$ has size at least $2^{\mathrm{tw}(G)^{\Omega(1/d)}}$ using the Grid Minor Theorem and Håstad's lower bound. The same technique can be directly applied to the resolution and it leads to the lower bound $S(\mathrm{T}(G, c)) \geq 2^{\Omega(\mathrm{tw}(G)^\delta)}$.

Our contributions. In this paper we prove a stronger lower bound for regular resolution, namely for an arbitrary graph $G(V, E)$,

$$S_R(\mathrm{T}(G, c)) \geq 2^{\Omega(\mathrm{tw}(G)/\log |V|)}.$$

For constant degree graphs this bound is tight up to a $\log |V|$ factor in the exponent.

We propose a new method of proving lower bound on the size of a regular resolution refutation of a Tseitin formula. The method is based on a transformation of a regular resolution proof of an unsatisfiable Tseitin formula into a read-once branching program (1-BP) computing a *satisfiable* Tseitin formula based on the same graph. Namely, we show that if there exists a regular resolution refutation of an unsatisfiable $\mathrm{T}(G, c)$ of size S then there exists a 1-BP computing a *satisfiable* Tseitin formula $\mathrm{T}(G, c')$ of size $S^{\mathcal{O}(\log n)}$ where n is the number of vertices in G. Using the similar idea we show how to transform a tree-like resolution refutation of an *unsatisfiable* Tseitin formula $\mathrm{T}(G, c)$ of size S into a 1-BP computing a *satisfiable* Tseitin formula $\mathrm{T}(G, c')$ of size $S + 1$.

The complexity of read-once branching programs computing satisfiable Tseitin formulas has been studied in [20, 14, 15]. Glinskih and Itsykson [14] proved that any read-once *nondeterministic* branching program (1-NBP) computing satisfiable Tseitin formula based on a spectral expander with n vertices has size at least $2^{\Omega(n)}$. In the more recent paper [15] Glinskih and Itsykson proved that the size of any 1-NBP computing a satisfiable Tseitin formula based on $n \times n$ grid is at least $2^{\Omega(n)}$. This result combined with the Grid Minor Theorem implies that any 1-NBP computing a satisfiable Tseitin formula $\mathrm{T}(G, c)$ has size at least $2^{\Omega(\mathrm{tw}(G)^{\delta})}$. It is also shown in [15] that every satisfiable $\mathrm{T}(G, c)$ can be computed by a 1-BP of size $2^{\mathcal{O}(\mathrm{tw}(G)\log|V|)}$. In this paper we show a stronger lower bound $2^{\Omega(\mathrm{tw}(G))}$ on the size of a nondeterministic read-once branching program computing a satisfiable $\mathrm{T}(G, c)$. In our proof we explicitly construct a tree decomposition of G given a 1-NBP computing $\mathrm{T}(G, c)$. In order to do it we introduce a new graph measure, the component width. On the one hand, the component width of a graph G is very close to the logarithm of the size of the smallest 1-NBP computing a satisfiable formula $\mathrm{T}(G, c)$, on the other hand, we will show that the component width of G is, roughly speaking, between the treewidth and the pathwidth of G.

We also show that there exists a family of constant degree graphs $G_n(V_n, E_n)$ such that any 1-NBP computing satisfiable $\mathrm{T}(G_n, c)$ has size at least $2^{\Omega(\mathrm{tw}(G_n)\log|V_n|)}$. This example implies the following:

- The upper bound $2^{\mathcal{O}(\mathrm{tw}(G)\log|V|)}$ on the size of 1-BP computing satisfiable $\mathrm{T}(G, c)$ proven in [15] can not be improved.
- It is impossible to eliminate $\log|V|$ factor from the exponent in the transformation of regular resolution proofs of an unsatisfiable Tseitin formula to a read-once branching program computing a satisfiable formula. Here we use the mentioned upper bound $S_R(\mathrm{T}(G_n, c')) = 2^{\mathcal{O}(\mathrm{tw}(G_n))}\mathrm{poly}(|V_n|)$ [1].
- The upper bound $S_T(\mathrm{T}(G, c')) \le 2^{\mathcal{O}(\mathrm{tw}(G)\log|V|)}$ from [3, 21] can not be improved. Here we use that tree-like resolution of $\mathrm{T}(G, c')$ of size S can be transformed to a 1-BP for satisfiable $\mathrm{T}(G, c)$ of size $S + 1$.
- Since $S_T(\mathrm{T}(G_n, c')) = 2^{\Omega(\mathrm{tw}(G_n)\log|V|)}$ and $S_R(\mathrm{T}(G_n, c')) \le 2^{\mathrm{tw}(G_n)}\mathrm{poly}(|V_n|)$, regular and tree-like resolutions may be superpolynomially separated on Tseitin formulas.

Organization of the paper. In Section 2 we give the basic definitions, preliminaries and detailed descriptions of our contribution. In Section 3 we describe the transformation of a regular resolution proof to a 1-BP computing a satisfiable Tseitin formula. In Section 4 we prove the lower bound for a 1-NBP computing a satisfiable Tseitin formula. The construction of graphs G_n is given in Subsection 4.4.

2 Preliminaries and results

Basic graph notation. Throughout the paper, we consider undirected graphs with no self-loops but possibly with parallel edges. We use $G(V, E)$ to denote a graph G with a vertex set V and an edge set E. By a *connected component* of a graph G we mean an inclusion-wise maximal connected subgraph of G. For example, it can be denoted as $C(U, E_U)$, where $U \subseteq V(G)$. By $\#G$ we denote the number of connected components in G. For the maximum degree of a graph G, we use the standard notion $\Delta(G)$.

Resolution refutations. A resolution refutation of an unsatisfiable CNF formula φ is a sequence of clauses $C_1, C_2, \dots, C_s$ such that 1) C_s is the empty clause (identically false), 2) for all $i \in [s]$, the clause C_i is either a clause of φ, or can be obtained by the resolution rule from two clauses with lesser numbers, where the resolution rule allows to derive $A \vee B$ from $A \vee x$ and $B \vee \neg x$. A resolution refutation is *tree-like* if every derived clause can be used as a premise of the resolution rule at most once. A resolution refutation is *regular* if for every increasing sequence $1 \leq i_1 < i_2 < \dots < i_k \leq s$ such that for all $j \in \{2, \dots, k\}$ the clause C_{i_j} is obtained by the resolution rule applied to $C_{i_{j-1}}$ as one of the premises (let x_j denote the resolved variable), all variables x_j for $j \in \{2, \dots, k\}$ are distinct. The number s is the size of the resolution refutation. For an unsatisfiable CNF formula φ we denote by $S(\varphi)$ the minimum size of resolution refutations of φ, by $S_R(\varphi)$ the minimum size of regular resolution refutations of φ, and by $S_T(\varphi)$ the minimum size of tree-like resolution refutations of φ. The inequality $S_T(\varphi) \geq S_R(\varphi)$ is well-known and the inequality $S_R(\varphi) \geq S(\varphi)$ is straightforward.

The width of a clause is the number of literals in it. The width of a resolution refutation $C_1, C_2, \dots, C_s$ is the maximum width of C_i for $i \in [s]$. The resolution width of an unsatisfiable CNF formula φ is the minimum possible width among all its resolution refutations. We denote the resolution width of φ by $w(\varphi)$.

Theorem 1 (Size-width relation [5])**.** *Let φ be an unsatisfiable formula in k-CNF with n variables. Then*

- $S_T(\varphi) \geq 2^{w(\varphi)-k}$;
- $S(\varphi) \geq 2^{\Omega((w(\varphi)-k)^2/n)}$.

Tseitin formulas. Let $G(V, E)$ be a graph. Let $c : V \to \{0, 1\}$ be a *charge function.* A Tseitin formula $\mathrm{T}(G, c)$ depends on the propositional variables x_e for $e \in E$. For each vertex $v \in V$ we define the parity condition of v as $P_v := (\sum_{e \text{ is incident to } v} x_e \equiv c(v) \bmod 2)$. The Tseitin formula $\mathrm{T}(G, c)$ is the conjunction of parity conditions of all the vertices: $\bigwedge_{v \in V} P_v$. Tseitin formulas is represented in CNF as follows: we represent P_v in CNF in the canonical way for all $v \in V$.

In this paper we define a connected component of a graph G as an inclusion-wise maximal connected subgraph of G. Assume that G consists of connected components $H_1, H_2, \dots, H_t$. Then a Tseitin formula $\mathrm{T}(G, c)$ is equivalent to the conjunction $\bigwedge_{i=1}^{t} \mathrm{T}(H_i, c)$. In the last formula we abuse the notation since c is defined not only on the vertices of H_i and, thus, we implicitly use the corresponding restriction on the set of vertices.

Lemma 2 ([28])**.** *A Tseitin formula $\mathrm{T}(G, c)$ is satisfiable if and only if for every connected component $C(U, E_U)$ of the graph G, the condition $\sum_{u \in U} c(u) \equiv 0 \bmod 2$ holds.*

Theorem 3 ([1])**.** *Let $\mathrm{T}(G, c)$ be an unsatisfiable Tseitin formula. Then there exists a* regular *resolution refutation of $\mathrm{T}(G, c)$ of size at most $2^{\mathcal{O}(w(\mathrm{T}(G,c)))} \cdot |\mathrm{T}(G, c)|$.*

Theorem 4 ([3, 21])**.** *Let $\mathrm{T}(G, c)$ be an unsatisfiable Tseitin formula based on a graph $G(V, E)$. Then there exists a* tree-like *resolution refutation of $\mathrm{T}(G, c)$ of size at most $2^{\mathcal{O}(w(\mathrm{T}(G,c)) \log |V|)}$.*

Tree and path decompositions. A *tree decomposition* of an undirected graph $G(V, E)$ is a tree $T(V_T, E_T)$ such that for every vertex $u \in V_T$ there is a corresponding set $X_u \subseteq V$ and it satisfies the following properties:

1. The union of X_u for $u \in V_T$ equals V.
2. For every edge $(a, b) \in E$ there exists $u \in V_T$ such that $a, b \in X_u$.
3. If a vertex $a \in V$ is contained in the sets X_u and X_v for some $u, v \in V_T$, then it is also contained in X_w for all vertices w on the unique path between u and v in T.

The sets X_u are called *bags* of the tree decomposition. The *width* of a tree decomposition is the maximum bag size $|X_u|$ for $u \in V_T$ minus one. A *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, is the minimum width among all tree decompositions of the graph G.

A *path decomposition* of a graph G is a tree decomposition of G such that the underlying tree T is a simple path. A *pathwidth* of a graph G, denoted by $\mathrm{pw}(G)$, is the minimum width among all path decompositions of the graph G.

A *line graph* of a graph $G(V, E)$ is a graph $L(G)$ with the set of vertices E such that two different edges $e_1, e_2 \in E$ are connected in $L(G)$ iff they have a common endpoint.

Theorem 5 (Corollary 8 and Corollary 16 in the ECCC version of [13])**.** *Let $G(V, E)$ be a graph and $\mathrm{T}(G, c)$ be an unsatisfiable Tseitin formula. Then $w(\mathrm{T}(G, c)) = \max\{\mathrm{tw}(L(G)), \Delta(G)\}$.*

Proposition 6 (see [6, 2] for the upper bound and [17] for the lower bound)**.** *Let $G(V, E)$ be a graph. Then $\frac{1}{2}(\mathrm{tw}(G) + 1) - 1 \leq \mathrm{tw}(L(G)) \leq (\mathrm{tw}(G) + 1) \cdot \Delta(G) - 1$.*

Branching programs. A branching program is a representation of a function $f : \{0, 1\}^n \to K$, where K is a finite set. A branching program for the function $f(x_1, x_2, \ldots, x_n)$ is a directed acyclic graph with $|K|$ sinks, sinks are labeled with different elements of the set K, each of the remaining nodes is labeled with a variable from $\{x_1, x_2, \ldots, x_n\}$ and has exactly two outgoing edges, the first is labeled with 0, the second is labeled with 1. Each node v of a branching program computes a function $f_v : \{0, 1\}^n \to K$. For a $k \in K$, the sink s labeled with k, computes the function $f_s \equiv k$. Assume that a node v is labeled with x_i, the outgoing edge from v labeled with 0 ends in a node v_0 and the outgoing edge labeled with 1 ends in a node v_1. Then $f_v(x_1, \ldots, x_n)$ equals $f_{v_1}(x_1, \ldots, x_n)$ if $x_i = 1$ and equals $f_{v_0}(x_1, \ldots, x_n)$ if $x_i = 0$. The size of a branching program is the number of nodes in it.

It is usually assumed that a branching program has only one source, in that case we say that the branching program computes the function computed in its source. We refer to a sink labeled with $k \in K$ as *k-sink*. We say that a branching program with unique source is a *decision tree* if every node of it except sinks has at most one incoming edge.

We say that a branching program *computes a relation* $Q \subseteq \{0, 1\}^n \times K$ if it computes a function $f : \{0, 1\}^n \to K$ such that for every $x \in \{0, 1\}^n$ the condition $(x, f(x)) \in Q$ holds.

A branching program is (syntactic) *read-once* if every path in it contains at most one occurrences of each variable.

One of the important concepts in proof complexity is the search problem Search_φ based on an unsatisfiable CNF-formula φ: given the values of the variables of φ, find a clause of φ that is falsified by these values. In many cases one can reduce proving lower bounds for proof systems to proving lower bounds on computing Search_φ in a related model of computation.

Theorem 7 ([22]). *1. The length of the shortest tree-like resolution refutation of φ ($S_T(\varphi)$) equals the size of the smallest decision tree for* Search_φ.

2. The length of the shortest regular resolution refutation of φ ($S_R(\varphi)$) equals the size of the smallest read-once branching program computing Search_φ.

Main result. Our main result is the following theorem.

Theorem 8. *Let $\mathrm{T}(G,c)$ be an unsatisfiable Tseitin formula based on a graph $G(V,E)$. Then $S_R(\mathrm{T}(G,c)) \geq 2^{\Omega(\mathrm{tw}(G)/\log(|V|))}$.*

Theorem 5, Theorem 3 and Proposition 6 imply that for constant degree graphs the bound from Theorem 8 is tight up to a logarithmic factor in the exponent.

The proof of Theorem 8 can be divided into two parts, and each of them is of independent interest:

1. We show that a regular resolution proof of an unsatisfiable Tseitin formula $\mathrm{T}(G,c)$ of size S can be transformed to a 1-BP computing satisfiable Tseitin formula $\mathrm{T}(G,c')$ of size $S^{\mathcal{O}(\log|V|)}$.
2. We prove that the size of any 1-BP computing a satisfiable $\mathrm{T}(G,c')$ is $2^{\Omega(\mathrm{tw}(G))}$.

2.1 From unsatisfiable to satisfiable Tseitin formulas

In the first part we prove the following theorem.

Theorem 9. *Let $\mathrm{T}(G,c)$ be an unsatisfiable Tseitin formula. If there exists a regular resolution refutation of $\mathrm{T}(G,c)$ of size S, then for every c' such that $\mathrm{T}(G,c')$ is satisfiable, there exists a* 1-BP *computing $\mathrm{T}(G,c')$ of size $S^{\mathcal{O}(\log n)}$, where n is the number of vertices in G.*

2.1.1 Falsified vertex vs falsified clause

For a graph $G(V,E)$ and a charge function $c: V \to \{0,1\}$ we define a relation $\mathrm{SearchVertex}(G,c)$ consisting of the pairs (σ, v) where $\sigma : \{x_e \mid e \in E\} \to \{0,1\}$ and $v \in V$ such that $\sum_{e \text{ is incident to } v} \sigma(x_e) \not\equiv c(v) \bmod 2$. If a Tseitin formula $\mathrm{T}(G,c)$ is unsatisfiable then the relation $\mathrm{SearchVertex}(G,c)$ is total i.e. for every $\sigma : \{x_e \mid e \in E\} \to \{0,1\}$ there exists $v \in V$ such that $(\sigma, v) \in \mathrm{SearchVertex}(G,c)$. We consider this relation as the following search problem: given the values of the variables find a vertex with the parity condition violated.

The problem $\mathrm{SearchVertex}(G,c)$ differs from $\mathrm{Search}_{\mathrm{T}(G,c)}$ in the granularity of the encoding: in the first case we search for a vertex with violated parity condition and in the second case we search for a falsified clause from a CNF representation of a violated parity condition. The problem $\mathrm{SearchVertex}(G,c)$ is not harder than $\mathrm{Search}_{\mathrm{T}(G,c)}$ since given a falsified clause it is easy to find a vertex with violated parity condition. It is easy to see that for decision trees the problems $\mathrm{SearchVertex}(G,c)$ and $\mathrm{Search}_{\mathrm{T}(G,c)}$ are equivalent. However, 1-BP complexities of $\mathrm{SearchVertex}(G,c)$ and $\mathrm{Search}_{\mathrm{T}(G,c)}$ are different. We will prove the following proposition in Subsection 3.5.

Proposition (Proposition 31). *1. There is a graph G_n with $2n+1$ vertices and maximal degree $2n$ such that there is a* 1-BP *for* $\mathrm{SearchVertex}(G_n,c')$ *of size* $\mathrm{poly}(n)$ *but any* 1-BP *for* $\mathrm{Search}_{\mathrm{T}(G_n,c')}$ *has size at least 2^n.*

2. *Let $K_{\log n}$ be a complete graph on $\log n$ vertices. Then* $\mathrm{SearchVertex}(K_{\log n}, c')$ *has* 1-BP *of size* $\mathrm{poly}(n)$ *but any* 1-BP *for* $\mathrm{Search}_{\mathrm{T}(K_{\log n}, c')}$ *has size at least* $2^{\Omega(\log^2 n)}$.

We do not know how the complexity of these problems behave for constant degree graphs. We conjecture that $\mathrm{SearchVertex}(G, c)$ and $\mathrm{Search}_{\mathrm{T}(G,c)}$ have polynomially related 1-BP complexities. The following proposition (proved in Subsection 3.5), however, shows that this conjecture implies the stronger statement than Theorem 8.

Proposition (Proposition 32)**.** *Assume that for every d there exists a polynomial q_d such that for every graph G with degrees at most d if there exists a* 1-BP *computing* $\mathrm{SearchVertex}(G, c)$ *of size S, then there exists a* 1-BP *computing* $\mathrm{Search}_{\mathrm{T}(G,c)}$ *of size $q_d(S)$. Then for every constant-degree graph G, $S_R(\mathrm{T}(G, c)) \geq 2^{\Omega(w(\mathrm{T}(G,c')))}$.*

2.1.2 Well-structured BPs

The problem $\mathrm{SearchVertex}(G, c)$ looks more essential than the problem $\mathrm{Search}_{\mathrm{T}(G,c)}$ since the second problem is dependent on the particular encoding of the Tseitin formula. We will prove lower bound on 1-BP complexity of $\mathrm{SearchVertex}(G, c)$, and it will imply a lower bound on 1-BP complexity of $\mathrm{Search}_{\mathrm{T}(G,c)}$ and, thus, by Theorem 7, it will imply a lower bound on regular resolution refutations of $\mathrm{T}(G, c)$.

At first we show that the minimum-size read-once branching programs computing satisfiable $\mathrm{T}(G, c)$ and $\mathrm{SearchVertex}(G, c)$ have good structure. Namely every node of a 1-BP solves the same problem but for some other graphs and charge function. We define this structure below.

Definition 10. We say that a branching program D is a *well-structured* branching program computing satisfiable Tseitin formulas if the following conditions hold:

- D has two sinks: one labeled with 0 and one labeled with 1 (all the other nodes of D are called inner nodes);
- There exists a finite set of vertices V and a map μ defined on the set of the nodes of D except the 0-sink that maps a node s to a pair (G_s, c_s), where $G_s(V, E_s)$ is a graph on the set of vertices V and $c_s \colon V \to \{0, 1\}$ is a charge function such that the formula $\mathrm{T}(G_s, c_s)$ is satisfiable. Every node s except the sinks is labeled with a variable x_e, where $e \in E_s$.
- (Sink condition) $\mu(\text{1-sink}) = (G_\emptyset(V, \emptyset), \mathbf{0})$, where $G_\emptyset$ is the graph without edges and $\mathbf{0}$ is identically zero function.
- (Local condition) Let s be a node labeled with x_e and let s_i be the end of the i-labeled edge outgoing from s for $i \in \{0, 1\}$. Let c_0 be equal to c_s and c_1 be obtained from c_s by flipping the charges at the endpoints of the edge e.
  - If e is not a bridge of G_s, then $G_{s_0} = G_{s_1} = G - e$, $c_{s_0} = c_0$ and $c_{s_1} = c_1$.
  - If e is a bridge of G_s, let V_A be the set of vertices of a connected component of $G_s - e$ that has a vertex incident to e. Let $\gamma = \sum_{v \in V_A} c_s(v)$. (Since $\mathrm{T}(G_s, c_s)$ is satisfiable, then by Lemma 2, γ does not depend on the choice of the component V_A.)

    Then $G_{s_\gamma} = G - e$, $c_{s_\gamma} = c_\gamma$ and $s_{1-\gamma}$ is the 0-sink.

Assuming that D has the unique source r and $\mu(r) = (G, c)$, we verify in Proposition 16 that D computes $\mathrm{T}(G, c)$.

We also define well-structured branching programs computing SearchVertex.

Definition 11. Let $G(V, E)$ be a *connected* graph and $\mathrm{T}(G, c)$ be unsatisfiable. Let D is a branching program with the unique source. We say that D is a *well-structured* branching program computing $\mathrm{SearchVertex}(G, c)$ if the following conditions hold:

- D has exactly $|V|$ sinks and each of them is labeled with a distinct element of V (all other nodes of D are called inner nodes);
- There exists a map ν from the nodes of D that maps a node s to a pair (G_s, c_s), where $G_s(V_s, E_s)$ is a *connected* subgraph of G and $c_s \colon V_s \to \{0, 1\}$ is a charge function such that $\mathrm{T}(G_s, c_s)$ is unsatisfiable. Every node s except the sinks is labeled with a variable x_e for some edge $e \in E_s$. The source is mapped by ν to the pair (G, c).
- (Sink condition) The sink labeled with v is mapped by ν to a graph with a single vertex v and a charge function that equals 1 on v.
- (Local condition) Let node s be labeled with a variable x_e and let s_i be the end of the i-labeled edge outgoing from s for $i \in \{0, 1\}$. Let c_0 be equal to c_s and c_1 be obtained from c_s by flipping the charges of c at the endpoints of e.
  - If e is not a bridge of G_s, then $G_{s_0} = G_{s_1} = G - e$, $c_{s_0} = c_0$ and $c_{s_1} = c_1$.
  - If e is a bridge of G_s, then $G - e$ can be represented as the disjoint union of two connected subgraphs of G_s: $A(V_A, E_A)$ and $B(V_B, E_B)$. Let $\gamma = \sum_{v \in V_A} c_s(v)$. Then $G_{s_\gamma} = B$, c_{s_γ} equals c_γ restricted to V_B, $G_{s_{1-\gamma}} = A$ and $c_{s_{1-\gamma}}$ equals $c_{1-\gamma}$ restricted to V_A.

The following Proposition 16 shows the correctness of this definition (i.e. that D indeed computes $\mathrm{SearchVertex}(G, c)$).

Proposition (Proposition 16)**.** *1. If D is a well-structured branching program computing satisfiable Tseitin formulas then a) D is a* 1-BP *and b) each node s of D except the* 0*-sink computes* $\mathrm{T}(G_s, c_s)$*, where $(G_s, c_s) = \mu(s)$.*

2. If D is a well-structured branching program computing $\mathrm{SearchVertex}(G, c)$*, then a) D is a* 1-BP *and b) each node s of D computes* $\mathrm{SearchVertex}(G_s, c_s)$*, where $(G_s, c_s) = \nu(s)$. In particular the source of D computes* $\mathrm{SearchVertex}(G, c)$*.*

The following lemma is rather easy:

Lemma 12 (partial case of ([14], Claim 15))**.** *Let D be a minimal* 1-BP *computing a satisfiable Tseitin formula* $\mathrm{T}(G, c)$*. Then D is a well-structured branching program computing* $\mathrm{T}(G, c)$*.*

The similar lemma for SearchVertex is not so straightforward, we will prove it in Subsection 3.2.

We say that a read-once branching program D is *locally minimal* satisfying some property if for any non-sink node s and any its direct successor t, if all edges incoming to s we redirect to t and remove s, then the resulting read-once branching program D' does not satisfy the same property.

Lemma (Lemma 17)**.** *Let $G(V,E)$ be a connected graph, and let c be such that* $\mathrm{T}(G,c)$ *is unsatisfiable. Let D be a locally-minimal* 1-BP *computing* SearchVertex(G,c)*. Then D is a well-structured branching program computing* SearchVertex.

Using Lemma 17, we prove the following theorem in Subsection 3.3:

Theorem (Theorem 14)**.** *Let $G(V,E)$ be a connected graph and a Tseitin formula* $\mathrm{T}(G,c)$ *be satisfiable and* $\mathrm{T}(G,c')$ *be unsatisfiable. Assume that there exists a* 1-BP *computing* SearchVertex(G,c') *of size S. Then there exists a* 1-BP *computing* $\mathrm{T}(G,c)$ *of size at most* $S^{\mathcal{O}(\log|V|)}$.

Notice that Theorem 14 and Theorem 7 imply Theorem 9.

Let us sketch the proof idea of Theorem 14. By Lemma 17 we may assume that a 1-BP computing SearchVertex(G,c') is well structured. Since definitions of well-structured branching programs are similar for computing satisfiable Tseitin formulas and SearchVertex, we will transform one branching program to another by induction in the reverse topological order. The only essential difference between well-structured branching programs for two problems is the local condition in the case when e is a bridge of G_s. In this case it cause that we can not just transform branching program, we need replicate some nodes (since they may be needed several times), and this is the reason of the increment of the size.

In case when a 1-BP computing SearchVertex(G,c') is a decision tree, one could eliminate replications and in Subsection 3.4 we prove the following theorem.

Theorem (Theorem 29)**.** *Let $G(V,E)$ be a connected graph and a Tseitin formula* $\mathrm{T}(G,c)$ *be satisfiable and* $\mathrm{T}(G,c')$ *be unsatisfiable. Assume that there exists a decision tree computing* SearchVertex(G,c') *of size S. Then there exists a* 1-BP *computing* $\mathrm{T}(G,c)$ *of size at most $S+1$.*

One would expect that a decision tree computing SearchVertex(G,c') of size S may be converted to a decision tree computing satisfiable $\mathrm{T}(G,c)$ of size poly(S). However, we prove the following proposition in Subsection 3.4.

Proposition (Proposition 30)**.** *Let P_n be a path of length n with doubled edges between every pair of consecutive vertices. Then there is a decision tree of size $\mathcal{O}(n^2)$ computing* SearchVertex(P_n,c') *for unsatisfiable* $\mathrm{T}(P_n,c')$*, but every decision tree for a satisfiable formula* $\mathrm{T}(P_n,c)$ *has size at least* 2^n.

2.2 Lower bound on the size of 1-NBP computing satisfiable Tseitin formulas

Nondeterministic branching programs. A nondeterministic branching program (NBP) represents a Boolean function $f:\{0,1\}^n \to \{0,1\}$. A nondeterministic branching program for a function $f(x_1,x_2,\ldots,x_n)$ is a directed a cyclic graph with one source and two sinks labeled with 0 and 1, each of the remaining nodes is either labeled with a variable from $\{x_1,x_2,\ldots,x_n\}$ and has exactly two outgoing edges. The first edge is labeled with 0, the second is labeled with 1 or the node is a guessing node that is unlabeled and has two outgoing unlabeled edges. So nondeterministic branching programs have three types of nodes: guessing nodes, nodes labeled with a variable (we call them just labeled nodes) and two sinks; the source is either a guessing node or a labeled node. The value of every node is defined recursively. The value of sinks and nodes labeled with a variable is defined as in case of deterministic programs. Assume that a node v is a guessing node and v_0 and v_1 are two direct successors of v. Then $f_v(x_1,\ldots,x_n)$ equals $f_{v_1}(x_1,\ldots,x_n) \vee f_{v_0}(x_1,\ldots,x_n)$. Note that deterministic branching programs with binary outputs constitute a special case of nondeterministic branching programs.

A nondeterministic branching program is (syntactic) *read-once* (1-NBP) if every path in it contains at most one occurrence of each variable.

Ordered binary decision diagrams. Let π be a permutation of the set $\{1, \dots, n\}$ (an order). A π*-ordered (nondeterministic) binary decision diagram* or $\pi-$OBDD ($\pi-$NOBDD) is a 1-BP (1-NBP) such that on every path from the source to a sink variable $x_{\pi(i)}$ can not appear before $x_{\pi(j)}$ if $i > j$. A *(nondeterministic) ordered binary decision diagram* or OBDD (NOBDD) is a π-ordered (nondeterministic) binary decision diagram for some π.

Our goal is to prove the following theorem.

Theorem 13. *Any* 1-NBP *computing satisfiable Tseitin formula* $\mathrm{T}(G, c)$ *has size at least* $2^{\Omega(\mathrm{tw}(G))}$.

An OBDD is a particular case of 1-NBP, however in Subsection 4.1 we prove the following lemma.

Lemma (Lemma 39)**.** *The size of any* 1-NBP *computing a satisfiable* $\mathrm{T}(G, c)$ *is at least the minimal size of* OBDD *computing* $\mathrm{T}(G, c)$.

In order to estimate the size of an OBDD computing satisfiable $\mathrm{T}(G, c)$ we introduce a new graph measure, the component width. For a graph $G(V, E)$, we define a game between Alice and Bob: Alice has a graph G_A and Bob has a graph G_B, both these graphs are on the same set of vertices V, and at the start of the game G_A has no edges and G_B equals G. On each turn, Bob chooses an edge e of G_B, remove it from G_B and add it to G_A. The game ends when G_B has no more edges. At every moment in the game we compute the value $\#G_A + \#G_B$, in the beginning this value equals $|V| + \#G$, the goal of Bob is to prevent this value from becoming too small. We say that Bob pays Alice the difference between the initial value $|V| + \#G$ and the minimum value of $\#G_A + \#G_B$ that occurs during the game. The *component width* of G (we denote it by $\mathrm{compw}(G)$) is defined as the minimum amount that Bob can pay in this game.

In Subsection 4.1 we prove the following theorem.

Theorem (Theorem 41)**.** *The size of any* 1-NBP *computing a satisfiable* $\mathrm{T}(G, c)$ *is at least* $2^{\mathrm{compw}(G)}$*.*

By Lemma 39, it is sufficient to prove Theorem 41 for a minimal OBDD computing $\mathrm{T}(G, c)$. The order of variables in a minimal OBDD corresponds to the strategy of Bob in the game defining $\mathrm{compw}(G)$, and the number of nodes in a minimal OBDD on every level is precisely $2^{|V|+\#G-(\#G_A+\#G_B)}$ (see Section 4.1 for details). We also prove an upper bound:

Proposition (Proposition 42)**.** *There exists an* OBDD *computing a satisfiable formula* $\mathrm{T}(G, c)$ *based on* $G(V, E)$ *of size at most* $|E| \cdot 2^{\mathrm{compw}(G)} + 2$.

It was proved in [15] that for any satisfiable Tseitin formula $\mathrm{T}(G, c)$ there is an OBDD computing it that has at most $2^{\mathrm{pw}(G)+1}$ nodes on every level. It implies the following corollary.

Corollary (Corollary 45)**.** *For any graph* G*,* $\mathrm{compw}(G) \leq \mathrm{pw}(G) + 1$.

In Subsection 4.3 we prove the lower bound:

Theorem (Theorem 52). *For any graph G,* $\mathrm{compw}(G) \geq \frac{1}{2}(\mathrm{tw}(G) - 1)$.

The proof of Theorem 52 is based on an explicit construction of an appropriate tree decomposition based on a Bob's strategy.

Theorem 52 and Theorem 41 implies Theorem 13.

2.3 Component width can be close to pathwidth

In Subsection 4.4 we show that if a graph G has specific properties (it can be represented as a strong product with a complete graph), then the component width of G is $\Omega(\mathrm{pw}(G))$. Using this we prove the following theorem.

Theorem (Theorem 63). *There exists a family of constant-degree graphs G_m such that G_m has n vertices, where $n = \Omega(m^3)$ and $n = \mathcal{O}(m^4)$, $\mathrm{tw}(G_m) = \Theta(m)$, $\mathrm{pw}(G_m) = \Theta(m \log m)$ and $\mathrm{compw}(G_m) = \Theta(m \log m)$.*

The following corollary shows that it is impossible to eliminate logarithmic factor in Theorem 9.

Corollary (Corollary 64). *Let S be the size of the smallest* 1-BP *computing* $\mathrm{SearchVertex}(G_m, c'_m)$. *Then size of any* 1-BP *computing a satisfiable* $\mathrm{T}(G_m, c_m)$ *is at least* $S^{\Omega(\log m)}$.

The following corollary implies at first, that the upper bound in Theorem 4 can not be improved, at second, that tree like resolution does not simulates regular resolution on Tseitin formulas.

Corollary (Corollary 65). *Size of any decision tree computing* $\mathrm{SearchVertex}(G_m, c'_m)$ *is at least* $2^{\Omega(\mathrm{tw}(G_m) \log m)}$.

3 From unsatisfiable to satisfiable Tseitin formulas

In this section we prove the following theorem.

Theorem 14. *Let $G(V, E)$ be a connected graph and a Tseitin formula* $\mathrm{T}(G, c)$ *be satisfiable and* $\mathrm{T}(G, c')$ *be unsatisfiable. Assume that there exists a* 1-BP *computing* $\mathrm{SearchVertex}(G, c')$ *of size S. Then there exists a* 1-BP *computing* $\mathrm{T}(G, c)$ *of size at most* $S^{\mathcal{O}(\log |V|)}$.

In the next two subsections we study the structure of 1-BPs computing SearchVertex. In Subsection 3.3 we prove Theorem 14 itself, in Subsection 3.4 we prove the version of this theorem for decision trees and in Subsection 3.5 we compare complexities of searching falsified clause and falsified vertex.

3.1 Well-structured branching programs

Lemma 15. *The result of the substitution $x_e := b$ to $\mathrm{T}(G, c)$ where $b \in \{0, 1\}$ is a Tseitin formula $\mathrm{T}(G', c')$ where $G' = G - e$ and c' differs from c on the endpoints of the edge e by b and equals c for every other vertex.*

Proof. The proof is straightforward. □

Proposition 16. *1. If D is a well-structured branching program computing satisfiable Tseitin formulas then a) D is a* 1-BP *and b) each node s of D except the* 0*-sink computes* $\mathrm{T}(G_s, c_s)$*, where* $(G_s, c_s) = \mu(s)$*. 2. If D is a well-structured branching program computing* SearchVertex(G, c)*, then a) D is a* 1-BP *and b) each node s of D computes* SearchVertex(G_s, c_s)*, where* $(G_s, c_s) = \nu(s)$*. In particular the source of D computes* SearchVertex(G, c)*.*

Proof. a) The proof of read-once property is the same for the both cases. Assume that there exists a path from s to an inner node $t \neq s$. Let s be labeled with x_e and t with $x_{e'}$. Let us show that $e \neq e'$. It follows from the local condition that G_t is a subgraph of G_s and that e does not belong to G_t. Since e' is an edge of G_t, $e \neq e'$.

b) For the both cases the proof is by induction on the vertices of branching programs in the reverse topological order. The base case follows from the sink conditions.

Inductive step. Let a node s be labeled with x_e. If e is not a bridge of G_s, then the local condition for s and the inductive hypothesis for the direct successors of s imply the statement for s.

Assume now that e is a bridge of G_s. Let s_0 and s_1 be the direct successors of s, where s_i is the endpoint of i-labeled edge outgoing from s.

1. The case of satisfiable Tseitin formulas. Let V_A be a set of vertices of a connected component of $G_s - e$ that have common vertex with e and let $\gamma = \sum_{v \in V_A} c_s(v)$. Let σ be a satisfying assignment of $\mathrm{T}(G_s, c_s)$. Consider the sum $\sum_{v \in V_A} \sum_{j \in E_s(v)} \sigma(x_j)$, where $E_s(v)$ is the set of all edges of G_s adjacent with v. Since σ satisfies $\mathrm{T}(G_s, c_s)$, this sum equals $\sum_{v \in V_A} c_s(v) = \gamma$; from the other hand, this sum equals $\sigma(x_e)$ since $\sigma(x_e)$ appears in this sum once while all other variables appears twice. Hence, there are no satisfying assignments of $\mathrm{T}(G_s, c_s)$ assigning x_e to $1 - \gamma$. By the inductive hypothesis, the node s_γ computes $\mathrm{T}(G_{s_\gamma}, c_{s_\gamma})$, where $G_{s_\gamma} = G_s - e$ and c_{s_a} differs from c_s on γ in the endpoints of e, hence s computes $\mathrm{T}(G_s, c_s)$.

2. The case of SearchVertex. By the inductive hypothesis s computes SearchVertex(G_{s_0}, c_{s_0}) if $x_e = 0$ and SearchVertex(G_{s_1}, c_{s_1}) if $x_e = 1$. Consider some assignment σ and let s returns a vertex v on σ. Denote $a := \sigma(x_e)$, then v is a vertex of G_{s_a}. If v is not incident to e, then σ falsifies the parity condition of $\mathrm{T}(G_s, c_s)$ of the vertex v since the sets of edges of G_s and G_{s_a} that are incident to v coincide. If v is incident to e, let us consider the sum $\sum_{j \in E_{s_a}(v)} \sigma(x_j)$, where $E_{s_a}(v)$ is the set of all edges incident to v in G_{s_a}. This sum equals $1 + c_{s_a}(v)$ since by the inductive hypothesis the assignment σ falsifies the parity condition for the vertex v in $\mathrm{T}(G_{s_a}, c_{s_a})$. By the local condition, $c_{s_a}(v) = c_s(v) + a$. Hence, σ falsifies the parity condition for v in $\mathrm{T}(G_s, c_s)$.

□

3.2 The structure of 1-BP computing SearchVertex

Let $F(V, E)$ be an undirected (not necessary connected) graph and let H be a connected component of F. We say that H is *a satisfiable component* of a formula $\mathrm{T}(F, c)$ if the formula $\mathrm{T}(H, c)$ is satisfiable. Otherwise we say that H is *an unsatisfiable component* of the formula $\mathrm{T}(F, c)$.

In this subsection we prove the following lemma.

Lemma 17. *Let $G(V, E)$ be a connected graph, and let c be such that* $\mathrm{T}(G, c)$ *is unsatisfiable. Let D be a locally-minimal* 1-BP *computing* $\mathrm{SearchVertex}(G, c)$. *Then D is a well-structured branching program computing* $\mathrm{SearchVertex}(G, c)$.

Moreover, for every node s, $\nu(s) = (H, f)$ and for every partial assignment α corresponding to a path from the source of D to s, H is the only unsatisfiable component of the formula $\mathrm{T}(G, c)|_\alpha$ *and f is the restriction of the charge function of* $\mathrm{T}(G, c)|_\alpha$ *to the vertices of H.*

Let D be a 1-BP that computes $\mathrm{SearchVertex}(G, c)$, where $G(V, E)$ is a connected graph and $\mathrm{T}(G, c)$ is unsatisfiable. For any internal node s of D we denote by $h(s)$ the set of labels of sinks reachable from s. We denote by $P(s)$ the set of partial assignments corresponding to the paths from the source of D to s.

The plan of the proof of Lemma 17 is the following.

1. The crucial point of the proof is the focus on the set $h(s)$ defined above. Proposition 19 shows that for every node s all partial assignments from $P(s)$ change charges of vertices from $h(s)$ in the same way.
2. The main technical part of the proof is the statement that if D is locally minimal 1-BP, then for its every node s, for all $\alpha \in P(s)$, $\mathrm{T}(G, c)|_\alpha$ has the unique unsatisfiable connected component and its set of vertices is precisely $h(s)$. In order to prove this we prove intermediate statements: (a) Proposition 20 shows that if D is a locally minimal 1-BP, then every its node s is labeled with an edge incident to $h(s)$. (b) Proposition 22 shows that if D is a locally minimal 1-BP, then every its node s is labeled with an edge from an unsatisfiable component of $\mathrm{T}(G, c)|_\alpha$ that lies in $h(s)$ for all $\alpha \in P(s)$ (by Proposition 19 this component does not depend on α). (c) Proposition 24 shows that if D is a locally minimal 1-BP, then for every its node s, $h(s)$ is the union of one or several unsatisfiable components of $\mathrm{T}(G, c)|_\alpha$ for all $\alpha \in P(s)$. (d) Proposition 25 finishes the proof of this item.
3. We prove Lemma 17 using the results of items 1 and 2.

We will use the following lemma about Tseitin formulas.

Lemma 18 ([19], Lemma 2.3)**.** *Let $G(V, E)$ be a connected graph and let $c : V \to \{0, 1\}$ be a charge function. Let $U \subsetneq V$ and $\Phi = \bigwedge_{v \in U} P_v$ be the conjunction of the parity conditions for all vertices from U. Then Φ is satisfiable.*

Proposition 19. *Let s be an internal node of D. Let α_1 and α_2 be the assignments from $P(s)$. Then the following conditions hold: 1. For every edge $e \in E$ incident to a vertex in $h(s)$, α_1 assigns a value to the variable x_e iff α_2 does. 2. For every $v \in h(s)$ the charge of v in* $\mathrm{T}(G, c)|_{\alpha_1}$ *is equal to the charge of v in* $\mathrm{T}(G, c)|_{\alpha_2}$.

Proof. Consider a vertex $v \in h(s)$ and consider some sink t labeled with v that is reachable from s. Let β be a partial assignment corresponding to a path from s to t. Notice that the set of variables assigned by β does not intersect the set of variables assigned by α_i for $i \in \{1, 2\}$ since D is a 1-BP. Let us define $\rho_i = \alpha_i \cup \beta$ for $i \in \{1, 2\}$.

Both the assignments ρ_1 and ρ_2 falsify the vertex v. Thus, for every edge e incident to v the value for x_e is assigned by ρ_1 and by ρ_2. Thus α_1 and α_2 assign values to the same subset of variables among $\{x_e \mid e \text{ is incident to } v\}$ and the sums modulo 2 of the values assigned to these variables by α_1 and α_2 are the same. □

Proposition 20. *Let D be a* locally minimal 1-BP *that computes* $\mathrm{SearchVertex}(G, c)$. *Then any internal node s of D is labeled with an edge incident to $h(s)$.*

Proof. Assume that for an inner node s labeled with x_e the statement is false i.e. e connects two vertices outside $h(s)$. Let t_0 and t_1 be the direct successors of s such that the edge (s, t_i) is labeled with i. Let us modify D as follows: remove the edge (s, t_0) and contract the edge (s, t_1). We denote the result of the contraction by s' and label it with the label of t_1 in D. We claim that D' also computes $\mathrm{SearchVertex}(G, c)$. Consider a full assignment β. Let $\beta'(x_q) = \beta(x_q)$ for $q \neq e$ and $\beta'(x_e) = 1$.

If the path in D corresponding to β does not pass through s, then exactly the same path with the same labels is contained in D', thus $D'(\beta) = D(\beta)$. Hence, it is sufficient to consider the case where the path in D corresponding to β passes through the node s. In this case the path in D corresponding to β' passes through s as well, because among the nodes of any path from the source to s only s is labeled with x_e. Then $D(\beta') \in h(s)$. The edge e is not incident to any vertex from $h(s)$ thus e is not incident to the vertex $D(\beta')$. Since the vertex $D(\beta')$ is falsified by β' and e is not incident to $D(\beta')$, then $D(\beta')$ is falsified by β as well. By the construction of D', the equality $D(\beta') = D'(\beta)$ holds. Thus, β falsifies $D'(\beta)$. Therefore, D' correctly computes $\mathrm{SearchVertex}(G, c)$ and this is a contradiction with the local minimality of D. □

By Lemma 15, the result of the substitution of a value to a variable of a Tseitin formula is a Tseitin formula as well. For an arbitrary assignment α from $P(s)$ we denote by $G_{s,\alpha}$ and $c_{s,\alpha}$ a graph and a charge function such that $\mathrm{T}(G, c)|_\alpha$ is precisely $\mathrm{T}(G_{s,\alpha}, c_{s,\alpha})$.

Notice that if for some $\alpha \in P(s)$, C is an unsatisfiable component of $\mathrm{T}(G_{s,\alpha}, c_{s,\alpha})$ and all its vertices are contained in $h(s)$, then by Proposition 19, C is an unsatisfiable component with respect to all partial assignments from $P(s)$. Let $U(s)$ be the set of all unsatisfiable components of $\mathrm{T}(G_{s,\alpha}, c_{s,\alpha})$ contained in $h(s)$, where α is some partial assignment from $P(s)$. By the remark above $U(s)$ does not depend on α.

Definition 21. Consider some $\alpha \in P(s)$. Let $H(V_H, E_H)$ be a connected component of $G_{s,\alpha}$ that contains at least one vertex from $h(s)$. Then there are three possible cases (three types of a component H with respect to a node s and a partial assignment α): (1) $V_H \subseteq h(s)$ and H is unsatisfiable connected component of $\mathrm{T}(G_{s,\alpha}, c_{s,\alpha})$. I.e. $H \in U(s)$; (2) $V_H \subseteq h(s)$ and H is satisfiable connected component of $\mathrm{T}(G_{s,\alpha}, c_{s,\alpha})$; (3) $V_H \not\subseteq h(s)$.

Proposition 22. *Let D be a* locally minimal 1-BP *that computes* $\mathrm{SearchVertex}(G, c)$. *Then any internal node s of D is labeled with a variable x_e, where e connects two vertices of a component from the set $U(s)$.*

Proof. By Proposition 20, we may assume that for every node l of D if l is labeled by x_e, then e is incident to a vertex of $h(l)$.

Assume that the statement of the proposition is false. Let us fix the deepest (i.e. the farthest from the source) node s of D violating the statement. Let s be labeled by x_e. Let t_0 and t_1 be the direct successors of s and the edge (s, t_i) be labeled with i for $i \in \{0, 1\}$. Let α be an assignment corresponding to some path from the source to s.

Since s violates the statement, e connects two vertices of a satisfiable component of $G_{s,\alpha}$ or connects two vertices of a component containing a vertex outside $h(s)$.

Let $C(V_C, E_C)$ be the connected component of $G_{s,\alpha}$ containing the edge e.

Let us consider an arbitrary partial assignment θ which satisfies all vertices from $V_C \cap h(s)$ and does not assign any variable corresponding to an edge of $G_{s,\alpha}$ outside C (if C is satisfiable, θ exists by definition and if C contains a vertex from the outside of $h(s)$, θ exists by Lemma 18).

Claim 23. *Consider a path $\tau = (\tau_1, \ldots, \tau_m)$ in D from $t_{\theta(x_e)}$ to a sink node. Then for every label $x_{e'}$ of a node of τ, e' is not incident to a vertex from C.*

Proof. Assume for the sake of contradiction that there exists a node violating the statement. Let $i \in [m]$ be the smallest index such that τ_i is labeled by $x_{e'}$ and e' is incident to a vertex from C.

Since τ_i is a successor of s, $h(\tau_i) \subseteq h(s)$. By Proposition 20, the edge e' is incident to $h(\tau_i)$. We are going to show that e' is not contained in an unsatisfiable component inside $h(\tau_i)$ and get a contradiction with the assumption that s is the deepest node violating the statement of the proposition.

Assume that e' is contained in an unsatisfiable component $C' \subseteq h(\tau_i)$. As it was mentioned before, the structure of such components is independent of a path from the source to τ_i, thus we choose a path that agrees with α on the path from the source to s and then continues as $\tau_1, \ldots, \tau_i$. Let μ be the partial assignment corresponding to this path. μ extends α, hence the graph $G_{\tau_i,\mu}$ is a subgraph of $G_{s,\alpha}$. C is a connected component of $G_{s,\alpha}$. C and C' has a common edge e'. Thus C' is a subgraph of C. The assignment θ satisfies the Tseitin formula corresponding to the connected component C and the charge function $c_{s,\alpha}$. Moreover, θ and μ have only one common variable in their domains. That variable is x_e and μ agrees with θ on x_e by the construction. Therefore, μ has a full extension that agrees with θ. But that extension satisfies all parity conditions of the vertices from C' which contradicts unsatisfiability of C' with respect to τ_i. □

The remaining part of the proof is similar to the proof of Proposition 20. Consider a diagram D' obtained by the removing the edge $(s, t_{1-\theta(e)})$ and the contraction of the edge $(s, t_{\theta(e)})$, where t_i is as before. Since D is a locally minimal 1-BP computing $\mathrm{SearchVertex}(G, c)$, there exists a full assignment β such that the vertex $D'(\beta)$ is not falsified by β. The path in D' corresponding to β passes through s since otherwise $D'(\beta) = D(\beta)$ which is falsified by β.

Let $\beta'(x_q) = \beta(x_q)$ for $q \neq e$, $\beta'(x_e) = \theta(x_e)$ and $v = D'(\beta)$.

As in the proof of Proposition 20, $D(\beta') = D'(\beta) = v$ and by the correctness of D, β' falsifies v. On the other hand, β does not falsify v by the choice of β. Since β' and β differ only on e, e is incident to v.

Since v is incident to e and e connects two vertices from C, $v \in C$. By Claim 23, the part of β' that corresponds to the path from $t_{\theta(x_e)}$ to a sink does not substitute values to edges that are incident to C, and since β' falsifies v, we get that v is a leaf in G_s. But the value $\beta'(x_e)$ was chosen according to the assignment θ satisfying all vertices in $V_C \cap h(s) \ni v$ and, thus, β' satisfies v that leads to a contradiction. □

Proposition 24. *Let D be a* locally minimal 1-BP *computing* $\mathrm{SearchVertex}(G, c)$*, where* $\mathrm{T}(G, c)$ *is unsatisfiable. Let s be a node of D. Let α be a partial assignment from the set $P(s)$. Then each vertex v from $h(s)$ is contained in an unsatisfiable component of* $\mathrm{T}(G_{s,\alpha}, c_{s,\alpha})$ *which is contained in $h(s)$.*

Proof. We prove the proposition by induction on the distance d from s to the furthest sink reachable from s.

Base case: $d = 0$, i.e. s is a sink. $h(s)$ consists of the only vertex v, the parity condition of v is falsified by the assignment α, then the component $\{v\}$ is unsatisfiable.

Induction step. Assume for the sake of contradiction that v is a vertex from $h(s)$ contained in a connected component $C(V_C, E_C)$ of type (2) or (3) (see Definition 21) with respect to the node s and the assignment $\alpha \in P(s)$. Let t_0 and t_1 be the direct successors of the node s. Notice that

$h(s) = h(t_0) \cup h(t_1)$ by the definition of h, thus there exists $i \in \{0, 1\}$ such that $v \in h(t_i)$. Consider $\beta_i \in P(t_i)$ that extends α.

Let s be labeled with a variable x_e; by Proposition 22, the edge e is contained in an unsatisfiable component of $\mathrm{T}(G, c)|_\alpha$, thus, e is not contained in the component C. If $V_C \setminus h(t_i) \neq \varnothing$, then v belongs to a connected component of type (3) with respect to the node t_i and the assignment β_i. If $V_C \subseteq h(t_i)$, then, since e is not contained in C, the connected component C equals the corresponding connected component of $\mathrm{T}(G, c)|_{\beta_i}$ contained in $h(t_i)$, moreover, the charges of the vertices of C are the same in the formulas $\mathrm{T}(G, c)|_{\beta_i}$ and $\mathrm{T}(G, c)|_\alpha$. Thus, in this case v is contained in a component of type (2) with respect to the node t_i and the assignment β_i.

But all vertices of $h(t_i)$ are contained in components of type (1) with respect to the node t_i and the assignment β_i by the induction hypothesis. This is a contradiction, hence all the vertices of $h(s)$ are contained in unsatisfiable components. $\square$

Proposition 25. *Let D be a* locally minimal 1-BP *computing* SearchVertex(G, c)*, where* $\mathrm{T}(G, c)$ *is unsatisfiable and G is connected. Let s be a node of D. Then $U(s)$ consists of a single connected component with the set of vertices $h(s)$. Moreover, for every $\alpha \in P(s)$, the single component from $U(s)$ is the only unsatisfiable component of* $\mathrm{T}(G, c)|_\alpha$.

Proof. We prove the statement by induction on the distance d from the source to s.

Base case: $d = 0$, i.e. s is the source of D. Since G is connected, it consists of the only unsatisfiable component of $\mathrm{T}(G, c)$. Lemma 18 implies that for every vertex $v \in V$ there exists a full assignment such that the parity condition of v is violated, but the parity condition of any other vertex is satisfied. Therefore, $h(s) = V$.

Induction step. Let α be a partial assignment from $P(s)$. Let r be the direct predecessor of s according to the path corresponding to α. Let $\beta \in P(r)$ agree with α. By the induction hypothesis, $U(r)$ consists of a single connected component $C(V_C, H_C)$ of the formula $\mathrm{T}(G, c)|_\beta$ with the vertex set $h(r)$. Let x_e be the label of the node r. By Proposition 22, the edge e is contained in C. We consider the following two cases.

If e is not a bridge of C, then for the substitution $x_e \coloneqq \alpha(x_e)$ to $\mathrm{T}(G, c)|_\beta$ the resulting formula has the only unsatisfiable component $C - e$. Lemma 18 implies that every vertex of V_C can be the only vertex, where the parity condition of $\mathrm{T}(G, c)|_\alpha$ is violated. Therefore, $h(s) = h(r) = V_C$.

Assume that e is a bridge of C. Let A, B be the connected components of $C - e$. The result of the substitution $x_e \coloneqq \alpha(x_e)$ to $\mathrm{T}(G, c)|_\beta$ (which is $\mathrm{T}(G, c)|_\alpha$) has the connected components A and B instead of C. Lemma 2 implies that exactly one of the components A and B is unsatisfiable. W.l.o.g. we assume that A is unsatisfiable component of $\mathrm{T}(G, c)|_\alpha$. By Lemma 18, every vertex of A can be the only vertex with violated parity condition, thus $h(s)$ contains all vertices of A. $h(s)$ does not contain any vertex of B since by Proposition 24 it can only contain vertices of unsatisfiable components. A is the single unsatisfiable component of $\mathrm{T}(G, c)|_\alpha$ since the substitution of any value to x_e in $\mathrm{T}(G, c)|_\beta$ does not affect any component except C. So, the induction step is proved. $\square$

Proof of Lemma 17. Consider an arbitrary node s of D. By Proposition 25 for every $\alpha \in P(s)$ a Tseitin formula $\mathrm{T}(G_{s,\alpha}, c_{s,\alpha})$ which is the result of substitution α to $\mathrm{T}(G, c)$ has the only unsatisfiable component $H \in U(s)$ with the set of vertices $h(s)$. Moreover, by Proposition 19, H and the restriction of $c_{s,\alpha}$ to the vertices of H does not depend on the choice of α. We denote by f the restriction of $c_{s,\alpha}$ to $h(s)$.

Fix $\alpha \in P(s)$ and consider an arbitrary path from s to a sink in D. Let θ be the partial assignment corresponding to this path. Let γ be the union of θ and α. Let $v = D(\gamma)$. Then γ falsifies the parity condition in the vertex v of $\mathrm{T}(G, c)$. Thus, θ falsifies the parity condition in the vertex v of the formula $\mathrm{T}(H, f)$ as well. Therefore the node s of D computes $\mathrm{SearchVertex}(H, f)$.

Let us verify that D is a well-structured branching program. We define $\nu(s) = (H, f)$, H is a connected subgraph of G and $\mathrm{T}(H, f)$ is unsatisfiable. If s is labeled with x_e, when by Proposition 22, e is an edge of H. Sink conditions are trivially satisfied. Local conditions can be verified in a straightforward manner since we know what is computed in every node of D. □

3.3 Proof of Theorem 14

Proposition 26. *Let $G(V, E)$ be a connected graph and let $c_1, c_2 : V \to \{0, 1\}$ be charge functions. If Tseitin formulas $\mathrm{T}(G, c_1)$ and $\mathrm{T}(G, c_2)$ are both satisfiable or both unsatisfiable, then one of them can be obtained from another by replacing some variables with their negations.*

Proof. A replacement x_e with $\neg x_e$ in a Tseitin formula corresponds to the flipping of the charges of the endpoints of the edge e. Since G is connected and $\mathrm{T}(G, c_1)$ and $\mathrm{T}(G, c_2)$ are both satisfiable or both unsatisfiable, then by Lemma 2 the charge functions c_1 and c_2 have even number of differences. Let $v_1, v_2, \ldots, v_{2k}$ be the vertices where c_1 differs from c_2. Let p_i be a simple path connecting v_{2i-1} and v_{2i} for $i \in [k]$. Let us modify $\mathrm{T}(G, c_1)$ in the following way: for each of the paths $p_1, \ldots, p_k$ we replace the variables corresponding to the edges of a path with their negations (if several paths pass through an edge e we will replace x_e with its negation as many times as the number of paths that pass through e). The resulting formula is $\mathrm{T}(G, c_2)$ since charges of the ends of the paths (i.e. in the vertices $v_1, \ldots, v_{2k}$) have been changed and charges of all other vertices have not been changed. □

Proof of Theorem 14. Let D be a minimum-size 1-BP computing $\mathrm{SearchVertex}(G, c)$ and let S be its size. By Lemma 18, every vertex of G can be the unique unsatisfied vertex of $\mathrm{T}(G, c)$, hence D contains at least $|V|$ sinks and, thus, $|S| \geq |V|$. By Lemma 17, D is a *well-structured* branching program computing $\mathrm{SearchVertex}(G, c)$.

By the item (1) of Proposition 16 and by Proposition 26, it is sufficient to construct a well-structured 1-BP computing a satisfiable $\mathrm{T}(G, c')$ of size $S^{\mathcal{O}(\log |V|)}$.

If V_H is a subset of V, then for a graph $H(V_H, E_H)$ we denote by $\widehat{H}(V, E_H)$ a graph that is obtained from H by adding isolated vertices $V \setminus V_H$. For a charge function $c_H : V_H \to \{0, 1\}$ we denote by $\widehat{c_H}$ a charge function $V \to \{0, 1\}$ that extends c_H to V by zero. For a vertex $w \in V$ we denote by $\mathbf{1}_w : V \to \{0, 1\}$ the charge function that equals 1 only on vertex w.

Enumerate the nodes of D in a reverse topological order $u_1, u_2, \ldots, u_S$, i.e. such that every edge of D is directed from a node with the greater number to a node with the less number. We assume that $\nu(u_i) = (G_i(V_i, E_i), c_i)$ for all $i \in [S]$. For k from 0 to S we iteratively construct a well-structured branching program $D^{(k)}$ computing a satisfiable Tseitin formula such that for every $i \in [k]$, for every charge function $c'_i : V_i \to \{0, 1\}$ that differs from c_i for exactly one vertex of V_i, there exists a node s of $D^{(k)}$ such that $\mu(s) = (\widehat{G_i}, \widehat{c'_i})$.

For $k = 0$, the program $D^{(0)}$ consists of the 0-sink and the 1-sink and $\mu(\text{1-sink}) = (G_\emptyset(V, \emptyset), \mathbf{0})$.

Assume that $D^{(k-1)}$ is constructed. We show how to add several nodes to $D^{(k-1)}$ and define μ for them such that the resulting program $D^{(k)}$ will be a correct well-structured branching program computing satisfiable Tseitin formulas satisfying the conditions for u_k.

If u_k is a sink labeled with a vertex v, then the graph G_k consists of the only vertex v and $c_k(v) = 1$. In that case we do not need to add any nodes to $D^{(k-1)}$ since the 1-sink satisfies the conditions for u_k.

Now assume that u_k is a non-sink node labeled with a variable x_e. Let the edge outgoing from u_k labeled with 0 end in a node u_{k_0} and the other edge outgoing from u_k end in a node u_{k_1}. For every vertex w of the graph G_k we will add a node s_w to $D^{(k)}$ and extends μ such that $\mu(s_w) = (\widehat{G_k}, \widehat{c_k} + \mathbf{1}_w)$,

We consider two cases:

(1): e is not a bridge of G_k. The local condition for D implies that the graphs G_{k_0} and G_{k_1} are equal to $G_k - e$. Let w be a vertex G_k, then it is a vertex of G_{k_0} and G_{k_1}. By the induction hypothesis for k_0 and k_1 there exist such nodes s_w^0 and s_w^1 in $D^{(k-1)}$ such that $\mu(s_w^0) = (\widehat{G_{k_0}}, \widehat{c_{k_0}} + \mathbf{1}_w)$ and $\mu(s_w^1) = (\widehat{G_{k_1}}, \widehat{c_{k_1}} + \mathbf{1}_w)$. We add to $D^{(k)}$ a node s_w, we label it with x_e and add an edge (s_w, s_w^0) labeled with 0 and an edge (s_w, s_w^1) labeled with 1, we define $\mu(s_w) = (\widehat{G_k}, \widehat{c_k} + \mathbf{1}_w)$. Notice that by the local condition for D, c_{k_0} equals c_k, and c_{k_1} differs from c_k only in the endpoints of e, thus the same statement is true for the charge functions $\widehat{c_{k_0}}, \widehat{c_{k_1}}$ and $\widehat{c_k}$ with flipped value at the vertex w. Therefore the local condition is satisfied for the node s_w in $D^{(k)}$ as well.

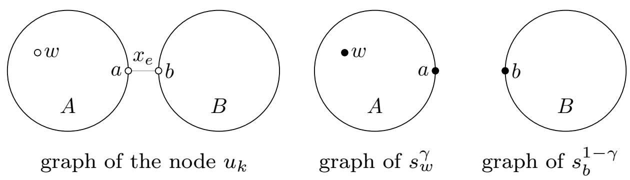

Figure 1: The graphs of the nodes (for $\gamma = 1$); nodes with the charge different from c_k are black, nodes with the same charge as in c_k are white.

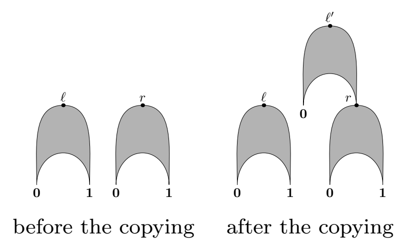

Figure 2: Copying.

(2): e is a bridge of G_k. Then $G_k - e$ can be represented as the disjoint union of two connected subgraphs of G_k: $A(V_A, E_A)$ and $B(V_B, E_B)$. Assume w.l.o.g. that A contains the vertex w. Let $a \in A$, $b \in B$ be the endpoints of the edge e.

Let $\gamma = \sum_{v \in V_A} c_k(v)$. We add to the program $D^{(k)}$ a node s_w labeled with x_e and define $\mu(s_w) = (\widehat{G_k}, \widehat{c_k} + \mathbf{1}_w)$, the edge outgoing from s_w labeled with γ ends in the 0-sink, and the edge labeled with γ will go to a node ℓ' such that $\mu(\ell') = ((\widehat{G_k - e}, \widehat{c_k} + \mathbf{1}_w + (1-\gamma)(\mathbf{1}_a + \mathbf{1}_b)))$. Local conditions for s_w will be satisfied, but we have to explain how to get such a node ℓ'.

By the local condition for D, $G_{k_{1-\gamma}} = A$ and $G_{k_\gamma} = B$. By the induction hypothesis $D^{(k-1)}$ contains a node $s_w^{1-\gamma}$ such that $\mu(s_w^{1-\gamma}) = (\widehat{A}, \widehat{c_{k_{1-\gamma}}} + \mathbf{1}_w)$ and a node s_b^γ such that $\mu(s_b^\gamma) = (\widehat{B}, \widehat{c_{k_\gamma}} + \mathbf{1}_b)$ (see Fig. 1).

Proposition 27. *Let V_H and V_F be two disjoint subsets of V, $h : V_H \to \{0,1\}$ and $f : V_F \to \{0,1\}$ be two charge functions and $H(V_H, E_H)$ and $F(V_F, E_F)$ be two graphs such that Tseitin formulas $\mathrm{T}(H,h)$ and $\mathrm{T}(F,f)$ are satisfiable. Let D_H and D_F supplied with mappings μ_H and μ_F be well-structured branching programs with disjoint set of nodes computing satisfiable Tseitin formulas $\mathrm{T}(\widehat{H}, \widehat{h})$ and $\mathrm{T}(\widehat{F}, \widehat{f})$. Consider a branching program $D_{H \cup F}$ which is obtained by redirecting edges of D_H going to the 1-sink to the source of the program D_F (and by deletions of 1-sink of D_H and merging two 0-sinks into a single 0-sink). Let us define a mapping $\mu_{H \cup F}$ defined on the nodes of*

$D_{H\cup G}$ *except the* 0*-sink as follows. If* s *is a node of* D_F*, we define* $\mu_{H\cup F}(s) = \mu_F(s)$*. If* s *is a node of* H_s *and* $\mu_H(s) = (H_s, c_s)$*, then we define* $\mu_{H\cup F}(s) = (H_s \cup F, c_s + \widehat{t})$*, where* $H_s \cup F$ *is a graph that is obtained from* H_s *by the addition of all edges from* E_F*. Then* $D_{H\cup F}$ *supplied with* $\mu_{H\cup F}$ *is a well-structured branching programs computing* $\mathrm{T}(\widehat{H \cup F}, \widehat{h} + \widehat{f})$*.*

Proof. Follows by the straightforward verification of local conditions. □

Proposition 27 explains how to create a node mapped by μ to $(\widehat{A \cup B}, \widehat{c_{k_{1-\gamma}}} + \mathbf{1}_w + \widehat{c_{k_\gamma}} + \mathbf{1}_b)$. Notice that by local properties of D, $\widehat{c_{k_{1-\gamma}}} + \widehat{c_{k_\gamma}} = \widehat{c_k} + (1-\gamma)\mathbf{1}_a + \gamma\mathbf{1}_b$, hence $\widehat{c_{k_{1-\gamma}}} + \mathbf{1}_w + \widehat{c_{k_\gamma}} + \mathbf{1}_b = \widehat{c_k} + \mathbf{1}_w + (1-\gamma)(\mathbf{1}_a + \mathbf{1}_b)$ and, thus, this node can be served as ℓ'. But the construction in Proposition 27 can not be used directly since it changes value of μ for several vertices and this can break the desired properties of $D^{(k)}$. So we have to copy several nodes.

If the number of vertices in the graph A is less or equal than the number of vertices in B, we denote $\ell = s_w^{1-\gamma}$ and $r = s_b^{\gamma}$, otherwise we denote $\ell = s_b^{\gamma}$ and $r = s_w^{1-\gamma}$. We copy the subprogram of ℓ (i.e. all successors of ℓ except the sinks) and add it to $D^{(k)}$. For every edge from the copied nodes to the 1-sink we redirect it to the node r. The edges to the 0-sink remain unchanged. We denote the source of the copied subprogram of ℓ by ℓ' (see Fig. 2). As we already discussed above, by Proposition 27, ℓ' satisfies all necessary properties.

Finally, we have that $D^{(S)}$ is a well-structured branching program computing satisfiable Tseitin formulas and contains a node s computing $\mathrm{T}(G, c')$, where c' differs from c in one vertex. We have to estimate the number of nodes in $D^{(k)}$. Notice that if we have two nodes with the same values of μ, then one of them may be deleted and all incoming edges may be redirected to the other. So we assume that all values of μ are different and we estimate the number of its possible values.

Claim 28. *Let* s *be a node of* $D^{(S)}$ *and* $\mu(s) = (Q, q)$*. Consider all connected components of* Q *with size at least two:* $C_1(V_{C_1}, E_{C_1}), C_2(V_{C_2}, E_{C_2}), \ldots, C_m(V_{C_m}, E_{C_m})$ *and assume that* $|V_{C_1}| \le |V_{C_2}| \le \cdots \le |V_{C_m}|$*. Then 1.* $|V_{C_i}| \ge |V_{C_1}| + \cdots + |V_{C_{i-1}}|$ *for every* $i \in [m]$*. 2. For every* $i \in [m]$ *there exists a node* s *of the program* D *such that* $\nu(s) = (C_i, h)$*, where* h *differs from* q *in exactly one vertex from* V_{C_i}*.*

Proof. We prove the statement by the induction on k for all nodes of $D^{(k)}$. $D^{(0)}$ consists of only sinks, hence the statement is true for $l = 0$. When we introduce the node s_w we always have that $\mu(s_w) = (\widehat{G_k}, \widehat{c_k} + \mathbf{1}_w)$, recall that $\nu(u_k) = (G_k, c_k)$ and, thus, $\widehat{G_k}$ has at most one connected component of size at least 2. And $\widehat{c_k} + \mathbf{1}_w$ differs from c_k on V only in w. The only remaining case where we introduce a node of $D^{(k)}$ is one when we copy nodes and then use the transformation from Proposition 27.

Consider the transformation from Proposition 27 applied for nodes ℓ and r of $D^{(k-1)}$. Let $\mu(r) = (\widehat{F}, \widehat{f})$, $\mu(\ell) = (\widehat{H}, \widehat{h})$, where $F(V_F, E_F)$, $H(V_H, E_H)$ are connected, $f : V_F \to \{0, 1\}$, $h : V_H \to \{0, 1\}$ and $V_F \cap V_H = \emptyset$. Consider a node t from the subprogram of ℓ. Since ℓ is in $D^{(k-1)}$, t is also in $D^{(k-1)}$. Let t' be the copy of t introduced during this step; let us verify the condition for t'. The local conditions of $D^{(k-1)}$ imply that $\mu(t) = (\widehat{H_t}, h_t)$, where H_t is a subgraph of H. The new node t' will have $\mu(t') = (H_{t'}, h_{t'})$, where $H_{t'}$ is obtained from $\widehat{H_t}$ by adding a new connected component $F(V_F, E_F)$, notice that all vertices of V_F are isolated in $\widehat{H_t}$. Notice that $|V_F| \ge |V_H|$; since H_t is a subgraph of H, $|V_F|$ is at least the total size of all connected components of H_t with at least two vertices. The function $h_{t'}$ differs from h_t only on V_F and coincides with f on V_F. Hence, the statement for t' follows from the inductive hypothesis for nodes r and t. □

Consider some node s of $D^{(S)}$, let $\mu(s) = (H, f)$. If v isolated vertex of H, then $f(v) = 0$. Assume that H consists of m connected components with at least two vertices. By the first item of Claim 28, $m \le \log|V|$. Consider some connected component $C(V_C, E_C)$ of H with at least two vertices. By the second item of Claim 28, there are at most $S|V|$ different values of the pair $(C, f|_{V_C})$. Hence, the number of different values of $\mu(s)$ is at most $\sum_{m=0}^{\lfloor \log|V| \rfloor} (|V|S)^m \le \log|V|(|V|S)^{\log|V|} = S^{\mathcal{O}(\log|V|)}$. $\square$

3.4 Case of decision tree

Theorem 29. *Let $G(V, E)$ be a connected graph and a Tseitin formula $\mathrm{T}(G, c)$ be satisfiable and $\mathrm{T}(G, c')$ be unsatisfiable. Assume that there exists a decision tree computing* $\mathrm{SearchVertex}(G, c')$ *of size S. Then there exists a* 1-BP *computing $\mathrm{T}(G, c)$ of size at most $S + 1$.*

Proof. The proof is a slight modification of the proof of Theorem 14. We use notations from that proof. Let T be a minimal decision tree computing $\mathrm{SearchVertex}(G, c)$ and let S be its size. By Lemma 17, T is a *well-structured* branching program computing $\mathrm{SearchVertex}(G, c)$.

We are going to construct a well-structured branching program computing a satisfiable $\mathrm{T}(G, c')$ of size at most $S + 1$. The theorem will follow by Proposition 16.

Enumerate the nodes of T in a reverse topological order $u_1, u_2, \ldots, u_S$. We assume that $\nu(u_i) = (G_i(V_i, E_i), c_i)$ for all $i \in [S]$.

By induction on k from 1 to S we show that for all $w \in V_k$ there exists a well-structured branching program $D_w^{(k)}$ with the only source s_w^k such that $\mu(s_w^k) = (\widehat{G_k}, \widehat{c_k} + \mathbf{1}_w)$ and the size of $D^{(k)}$ is at most the size of the subtree of u_k in T.

If u_k is a sink labeled with a vertex v, then $D_v^{(k-1)}$ consists of the 1-sink and $\mu(\text{1-sink}) = (G_\emptyset(V, \emptyset), \mathbf{0})$.

Now assume that u_k is a non-sink node labeled with a variable x_e. Let the 0-labeled edge outgoing from u_k end in a node u_{k_0} and the other edge outgoing from u_k end in a node u_{k_1}. For each $w \in V_k$ we are going to construct a well-structured branching program $D_w^{(k)}$ with the source s_w^k such that $\mu(s_w^k) = (\widehat{G_k}, \widehat{c_k} + \mathbf{1}_w)$.

We consider two cases:

(1): e is not a bridge of G_k. The local condition for T implies that the graphs G_{k_0} and G_{k_1} are equal to $G_k - e$. By the induction hypothesis there exist such well-structured branching programs $D_w^{(k_0)}$ and $D_w^{k_1}$ with sources $s_w^{k_0}$ and $s_w^{k_1}$ such that $\mu(s_w^{k_0})) = (\widehat{G_{k_0}}, \widehat{c_{k_0}} + \mathbf{1}_{w_k})$ and $\mu(s_w^{k_1}) = (\widehat{G_{k_1}}, \widehat{c_{k_1}} + \mathbf{1}_{w_k})$. Let us define $D_w^{(k)}$ as follows it has a source s_w^k, we label it with x_e and add an edge $(s_w^k, s_w^{k_0})$ labeled with 0 and an edge $(s_w^k, s_w^{k_1})$ labeled with 1, we define $\mu(s_w^k) = (\widehat{G_k}, \widehat{c_k} + \mathbf{1}_w)$. Notice that by the local condition for T, c_{k_0} equals c_k, and c_{k_1} differs from c_k only in the endpoints of e, thus the same statement is true for the charge functions $\widehat{c_{k_0}}, \widehat{c_{k_1}}$ and $\widehat{c_k}$ with flipped value at the vertex w. Therefore the local condition is satisfied for the node s_w^k in $D_w^{(k)}$ as well. And $|D_w^{(k)}| \le |D_w^{(k_1)}| + |D_w^{(k_0)}| + 1$, hence $|D_w^{(k)}|$ is at most the size of the subtree of u_k.

(2): e is a bridge of G_k. Then $G_k - e$ can be represented as the disjoint union of two connected subgraphs of G_k: $A(V_A, E_A)$ and $B(V_B, E_B)$. Assume w.l.o.g. that A contains the vertex w. Let $a \in A$, $b \in B$ be the endpoints of the edge e. Let $\gamma = \sum_{v \in V_A} c_k(v)$. We construct a program $D_w^{(k)}$ with a source s_w^k labeled with x_e and define $\mu(s_w^k) = (\widehat{G_k}, \widehat{c_k} + \mathbf{1}_w)$, the edge outgoing from s_w^k labeled with γ ends in the 0-sink, and the edge labeled with γ will go to a node ℓ such that

$\mu(\ell) = ((\widehat{G_k - e}, \widehat{c_k} + \mathbf{1}_w + (1-\gamma)(\mathbf{1}_a + \mathbf{1}_b)))$. Local conditions for s_w^k will be satisfied, but we have to explain how to get such a node ℓ.

By the local condition for T, $G_{k_{1-\gamma}} = A$ and $G_{k_\gamma} = B$. By the induction hypothesis there are a well-structured branching programs $D_w^{(k_{1-\gamma})}$ and $D_b^{(k_\gamma)}$ with the sources $s_w^{k_{1-\gamma}}$ and $s_b^{k_\gamma}$ such that $\mu(s(k_{1-\gamma}) = (\widehat{A}, \widehat{c_{k_{1-\gamma}}} + \mathbf{1}_w)$ and a node $s_b^{k_\gamma}$ such that $\mu(s_b^{k_\gamma}) = (\widehat{B}, \widehat{c_{k_\gamma}} + \mathbf{1}_b)$. We apply Proposition 27 to branching programs of $s(k_0)$ and $s(k_1)$ and denote the source of the resulting program by ℓ. By Proposition 27, $\mu(\ell) = (\widehat{A \cup B}, \widehat{c_{k_{1-\gamma}}} + \mathbf{1}_w + \widehat{c_{k_\gamma}} + \mathbf{1}_b)$. Notice that by local properties of T, $\widehat{c_{k_{1-\gamma}}} + \widehat{c_{k_\gamma}} = \widehat{c_k} + (1-\gamma)\mathbf{1}_a + \gamma \mathbf{1}_b$, hence $\widehat{c_{k_{1-\gamma}}} + \mathbf{1}_w + \widehat{c_{k_\gamma}} + \mathbf{1}_b = \widehat{c_k} + \mathbf{1}_{w_k} + (1-\gamma)(\mathbf{1}_a + \mathbf{1}_b)$.

$D_w^{(k)}$ is obtained from $D_w^{(k_{1-\gamma})}$ and $D_b^{(k_\gamma)}$, we also add its source and possibly add the 0-sink, but we delete one of 1-sinks, hence $|D_w^{(k)}| \leq |D_w^{(k_{1-\gamma})}| + |D_b^{(k_\gamma)}| + 1$, hence by induction hypothesis $|D_w^{(k)}|$ is at most the size of the subtree of u_k.

Consider a vertex $w \in V$, we have that $D_w^{(S)}$ is a well-structured branching program computing satisfiable Tseitin formulas computing $\mathrm{T}(G, c + \mathbf{1}_w)$ of size at most $S + 1$. ☐

Proposition 30. *Let P_n be a path of length n with doubled edges between every pair of the consecutive vertices. Then there is a decision tree of size $\mathcal{O}(n^2)$ computing* SearchVertex(P_n, c') *for unsatisfiable* $\mathrm{T}(P_n, c')$*, but every decision tree for a satisfiable formula* $\mathrm{T}(P_n, c)$ *has size at least* 2^n.

Proof. $\mathrm{T}(P_n, c)$ is satisfiable and has exactly 2^n satisfying assignments (an assignment satisfies $\mathrm{T}(P_n, c)$ iff parallel edges have the same values). Any two different accepting paths in a decision tree have different penultimate nodes. Every satisfying assignment of $\mathrm{T}(P_n, c)$ is realized by an accepting path, we mark the penultimate nodes on these paths. A Tseitin formula can not be satisfied by a partial assignment, hence no two satisfying assignments correspond to the same accepting path. Thus, any decision tree for $\mathrm{T}(P_n, c)$ has at least 2^n marked nodes.

We claim that SearchVertex(P_n, c') can be solved by a decision tree of size $\mathcal{O}(n^2)$. Indeed, we branch on the values of two central edges and for each of the four cases we get only one connected component that is unsatisfiable, so we will search for a falsified vertex in a graph of twice smaller size. The size of the resulting decision tree can be determined by the recurrence $S(n) = 4S(n/2)$, hence $S(n) = \mathcal{O}(n^2)$. ☐

3.5 Falsified vertex vs falsified clause

In this section we compare 1-BP complexity of SearchVertex(G, c) and $\mathrm{Search}_{\mathrm{T}(G,c)}$.

We show that SearchVertex(G, c) can be much easier than $Search_{\mathrm{T}(G,c)}$ for large and logarithmic degrees.

Proposition 31. *1. There is a graph G_n with $2n+1$ vertices and maximal degree $2n$ such that there is a* 1-BP *for* SearchVertex(G_n, c') *of size* poly(n) *but any* 1-BP *for* $\mathrm{Search}_{\mathrm{T}(G_n,c')}$ *has size at least* 2^n. *2. Let $K_{\log n}$ be a complete graph on $\log n$ vertices. Then* SearchVertex$(K_{\log n}, c')$ *has* 1-BP *of size* poly(n) *but any* 1-BP *for* $\mathrm{Search}_{\mathrm{T}(K_{\log n},c')}$ *has size at least* $2^{\Omega(\log^2 n)}$.

Proof. 1. Consider a graph G_n that consists of n triangles a_i, b_i, v for $i \in [n]$ sharing the common vertex v. It is easy to see that there is a 1-BP for SearchVertex(G_n, c') of size $\mathcal{O}(n)$. Indeed, we query edges of the first triangle and check parity conditions of vertices a_1, b_1, then we query edges of the second triangle and etc. We also save the current parity of vertex v; in order to do it we create for all $i \in [n-1]$ two nodes of 1-BP corresponding to different values of the current parity.

On the other hand, there are at least 2^n clauses from the parity condition of v in $\mathrm{T}(G_n, c')$ that can be uniquely falsified, hence any 1-BP for $\mathrm{Search}_{\mathrm{T}(G,c')}$ has at least 2^n sinks.

2. Using the technique connecting the expansion of a graph with the resolution width of the corresponding Tseitin formula [5] it is easy to show that the length of the shortest resolution refutation of $\mathrm{T}(K_{\log n}, c')$ is $2^{\Omega(\log^2 n)}$ (see [8] for details), thus the size of any 1-BP for $\mathrm{Search}_{\mathrm{T}(K_{\log n},c')}$ is $2^{\Omega(\log^2 n)}$. On the other hand, the 1-BP complexity of $\mathrm{SearchVertex}(K_{\log n}, c')$ is $\mathrm{poly}(n)$: chose an arbitrary order on edges of $K_{\log n}$ and query variables according to the chosen order. We save the current parities for all vertices, since there are $\log n$ vertices, we need only n nodes on every level. ☐

We do not know what happens for constant degree graphs. We conjecture that $\mathrm{SearchVertex}(G, c)$ and $\mathrm{Search}_{\mathrm{T}(G,c)}$ have polynomially connected 1-BP complexities. The following proposition, however, shows that this conjecture implies the stronger statement than Theorem 8.

Proposition 32. *Assume that for every d there exists a polynomial q_d such that for every graph G with degrees at most d if there exists a* 1-BP *computing* $\mathrm{SearchVertex}(G, c)$ *of size S, then there exists a* 1-BP *computing* $\mathrm{Search}_{\mathrm{T}(G,c)}$ *of size $q_d(S)$. Then for every constant degree G, $S_R(\mathrm{T}(G, c)) \geq 2^{\Omega(w(\mathrm{T}(G,c')))}$.*

Proof. Recall that the xorification of CNF formula φ is a formula $\varphi^{\oplus}$ that can be obtained from φ as follows: 1) we substitute xor of two fresh variables instead of every variable and then 2) translate every substituted clause to CNF. Alekhnovich and Razborov noticed that for every unsatisfiable CNF formula φ, $S(\varphi^{\oplus}) \geq 2^{\Omega(w(\varphi))}$ (the proof of this result can be found in [4]).

Let a graph $G^{\oplus}$ be obtained from a graph $G(V, E)$ by the doubling of every edge in E (we add a parallel copy for every edge of G). Notice that the formula $\mathrm{T}(G^{\oplus}, c)$ is the xorification of $\mathrm{T}(G, c)$. Hence, $S(\mathrm{T}(G^{\oplus}, c)) \geq 2^{\Omega(w(\mathrm{T}(G,c)))}$, and, thus, $S_R(\mathrm{T}(G^{\oplus}, c)) \geq 2^{\Omega(w(\mathrm{T}(G,c)))}$. Since degrees of G are at most D, degrees of $G^{\oplus}$ are at most $2d$, hence any 1-BP computing $\mathrm{SearchVertex}(G^{\oplus}, c)$ has size at least $2^{\Omega(w(\mathrm{T}(G,c)))}$.

Notice that if there exists a read-once branching program D of size S computing $\mathrm{SearchVertex}(G, c)$, then there exists a read-once branching program $D^{\oplus}$ of size at most $3S$ for the problem $\mathrm{SearchVertex}(G^{\oplus}, c)$. Indeed, assume that for all $e \in E$ we add a parallel edge e'. Let us perform the following modification of D consequently for all edges $e \in E$: for every node s labeled with x_e we create two nodes s_0 and s_1 labeled with $x_{e'}$. If an edge (s, s') in D was labeled by a, we add two edges: (s_0, s') with label a and (s_1, s') with label $1 - a$. We also add two edges: from s to s_0 labeled with 0 and to s_1 labeled with 1.

Thus we get that size of any 1-BP for $\mathrm{SearchVertex}(G, c)$ is at least $2^{\Omega(w(\mathrm{T}(G,c)))}$. Hence, the size of any 1-BP computing $\mathrm{Search}_{\mathrm{T}(G,c)}$ is at least $2^{\Omega(w(\mathrm{T}(G,c)))}$. The statement now follows from Theorem 7. ☐

4 Lower bound on 1-NBP computing satisfiable Tseitin formulas

The goal of this section is to prove a lower bound on size of 1-NBP computing satisfiable Tseitin formulas. In Subsection 4.1 we show that the minimal 1-NBP computing satisfiable Tseitin formula is an OBDD, in Subsection 4.2 we define the notion of component width and connect it with the OBDD complexity of Tseitin formulas, in Subsection 4.3 we prove the lower bound on the component width via the treewidth and in Subsection 4.4 we show that the component width may be close to the pathwidth.

4.1 Minimal read-once branching program for satisfiable Tseitin formulas is OBDD

In this subsection we prove that for a satisfiable Tseitin formula $\mathrm{T}(G, c)$ the minimal size of a nondeterministic read-once branching program computing it is at least the minimal size of an OBDD computing $\mathrm{T}(G, c)$. We also introduce a new graph measure, the component width, that approximate the logarithm of the minimal size of an OBDD computing $\mathrm{T}(G, c)$.

Let for a graph G the number $\#G$ denote the number of connected components in G.

Lemma 33 ([14]). *If a Tseitin formula* $\mathrm{T}(G, c)$ *is satisfiable, then it has* $2^{|E|-|V|+\#G}$ *satisfiable assignments.*

Let $G(V, E)$ be a graph. We denote by $A_{G,c}$ the set of satisfying assignments of $\mathrm{T}(G, c)$.

For every $J \subseteq E$ and every $\alpha \in A_{G,c}$ we denote a partial assignment α_J that restricts α to the set of variables $\{x_e \mid e \in J\}$.

Let $\mathcal{F}_J(G, c)$ be the set of Boolean functions that can be obtained from $\mathrm{T}(G, c)$ by application of α_J for $\alpha \in A_{G,c}$: $\mathcal{F}_J(G, c) = \{\mathrm{T}(G, c)|_{\alpha_J} \mid \alpha \in A_{G,c}\}$. Notice that all functions from $\mathcal{F}_J(G, c)$ are obtained from the Tseitin formula by substitutions, hence, by Lemma 15, they can be represented by Tseitin formulas.

For every $J \subseteq E$ we denote by $G_J(V, J)$ a subgraph of G that is based on the set of edges J. The next proposition estimates the number of ways to get every function from $\mathcal{F}_J(G, c)$.

Proposition 34. *For every* $f \in \mathcal{F}_J(G, c)$ *the size of the set* $\{\alpha \in A_{G,c} \mid f = \mathrm{T}(G, c)|_{\alpha_J}\}$ *equals* $2^{|E|-2|V|+\#G_{E\setminus J}+\#G_J}$.

Proof. Consider an assignment $\beta \in A_{G,c}$ such that $f = T(G, c)|_{\beta_J}$. Notice that $\beta_{E\setminus J}$ satisfies f. Let c' be a charge function such that $f = \mathrm{T}(G_{E\setminus J}, c')$. Notice that β_J is a satisfying assignment of $\mathrm{T}(G_J, c + c')$.

Notice that if γ is a satisfying assignment of $\mathrm{T}(G_J, c + c')$ and δ is a satisfying assignment of f, then $\gamma \cup \delta$ is a satisfying assignment of $\mathrm{T}(G, c)$. Thus, the size of the set $\{\alpha \in A_{G,c} \mid f = \mathrm{T}(G, c)|_{\alpha_J}\}$ equals the product of the number of satisfying assignments of f and the number of satisfying assignments of $\mathrm{T}(G_J, c + c')$. By Lemma 33, the latter product equals $2^{|E|-2|V|+\#G_{E\setminus J}+\#G_J}$. □

Remark 35. *Let D be an arbitrary* 1-NBP *computing a satisfiable $T(G, c)$. Since every satisfying assignment of* $\mathrm{T}(G, c)$ *assigns values to all variables, any accepting path (a path from the source of D to* 1*-sink) must contain all variables among labels of nodes.*

The following proposition was explicitly proved in [14].

Proposition 36. *Let D be a* 1-NBP *computing a satisfiable* $\mathrm{T}(G, c)$*. Let s be a node of D such that there is an accepting path passing through s. Then the following holds: 1) every two paths from the source to s assign values to the same set of variables $\{x_e \mid e \in J\}$, where $J \subseteq E$; 2) the maximal number of accepting paths passing through s corresponding to different satisfying assignments of* $\mathrm{T}(G, c)$ *is at most* $2^{|E|-2|V|+\#G_{E\setminus J}+\#G_J}$.

Proof. Consider an accepting path α passing through s. Let α^1 be a part of α from the source to s and α^2 — from s to 1-sink. Consider a path β^1 from the source to s. Notice that the path $\beta^1\alpha^2$ is also an accepting path. Since by Remark 35, every satisfying assignment of $\mathrm{T}(G,c)$ assigns values to all variables and every variable appears at most once on every path of D, sets of edges corresponding to variables assigned along α^1 and β^1 coincide; we denote this set of edges by J. Let for a path γ in D, ρ_γ denotes the partial assignment corresponding to γ. Both $\mathrm{T}(G,c)|_{\rho_{\alpha^1}}$ and $\mathrm{T}(G,c)|_{\rho_{\beta^1}}$ are Tseitin formulas based on the same graph $G-E$ and since they have common satisfying assignment ρ_{α^2}, they coincide. Hence, the number of accepting paths (corresponding to different assignments) passing through s does not exceed $2^{|E|-2|V|+\#G_{E\setminus J}+\#G_J}$ by Proposition 34. $\square$

For a graph $G(V,E)$ and a set of edges $J\subseteq E$ we introduce the notation

$$\mathrm{comp}_J(G)=|V|-\#G_{E\setminus J}-\#G_J+\#G.$$

Proposition 37. *The size of the set $\mathcal{F}_J(G,c)$ is equal to $2^{\mathrm{comp}_J(G)}$.*

Proof. By Lemma 33, the number of satisfying assignments of $\mathrm{T}(G,c)$ equals $2^{|E|-|V|+\#G}$. Every satisfying assignment of $\mathrm{T}(G,c)$ corresponds to a function from $\mathcal{F}_J(G,c)$; every function from $\mathcal{F}_J(G,c)$ corresponds to $2^{|E|-2|V|+\#G_{E\setminus J}+\#G_J}$ satisfying assignments of $\mathrm{T}(G,c)$ by Proposition 34. Hence, $\mathcal{F}_J(G,c)$ contains exactly $2^{|V|-\#G_{E\setminus J}-\#G_J+\#G}=2^{\mathrm{comp}_J(G)}$ elements. $\square$

Let D be a nondeterministic OBDD using the order of variables $x_{\pi(1)},x_{\pi(2)},\dots,x_{\pi(n)}$. For every $i\in[n]$, the i-th level of D is the set of nodes labeled with variable $x_{\pi(i)}$.

Lemma 38. *Let $\mathrm{T}(G,c)$ be a satisfiable Tseitin formula based on a graph $G(V,E)$. Let π be a permutation of $[|E|]$. For every i from 0 to $|E|$ we denote by J_i the set $\{e_{\pi(1)},e_{\pi(2)},\dots,e_{\pi(i)}\}$ of the first i edges according to permutation π. 1) Let D be a nondeterministic π-ordered OBDD computing $\mathrm{T}(G,c)$. Then for every i from $[|E|]$, the i-th level of D contains at least $2^{\mathrm{comp}_{J_{i-1}}(G)}$ nodes. 2) If D is a minimal π-OBDD computing $\mathrm{T}(G,c)$, then the i-th level of D has exactly $2^{\mathrm{comp}_{J_i}(G)}$ nodes and for every node s from the i-th level of D there are exactly $2^{|E|-2|V|+\#G_{E\setminus J_{i-1}}+\#G_{J_{i-1}}}$ accepting paths going through s.*

Proof. 1) By Remark 35, every accepting path of D contains a node from the i-th level. Since D is a NOBDD, for every path p from the source to a node from the i-th level, the set of labels of nodes from p is exactly $x_{e_{\pi(1)}},\dots,x_{e_{\pi(i)}}$. Hence, the i-th level of D should contain nodes computing all different functions that can be obtained from $\mathrm{T}(G,c)$ by the substitution of values of variables $x_{e_{\pi(1)}},\dots,x_{e_{\pi(i-1)}}$ according to a satisfying assignment of $\mathrm{T}(G,c)$. By Proposition 37, there are exactly $2^{\mathrm{comp}_{J_{i-1}}(G)}$ such functions.

2) Since D is a minimal π-OBDD, for every node s (except the 0-sink), there is an accepting path p passing through s. Let p correspond to a satisfying assignment α of $\mathrm{T}(G,c)$. Also by the minimality of D, there are no two nodes in the i-th level computing the same function, since otherwise these two nodes may be joined and it will decrease the size of D. Hence, by the first part of the proof, i-th level contains exactly $2^{\mathrm{comp}_{J_{i-1}}(G)}$ nodes.

Since D is a deterministic OBDD and by Remark 35 every accepting path of D contains all variables, every satisfying assignment of $\mathrm{T}(G,c)$ corresponds to exactly one accepting path in D. By Proposition 34, there are exactly $2^{|E|-2|V|+\#G_{E\setminus J_{i-1}}+\#G_{J_{i-1}}}$ satisfying assignments β of $\mathrm{T}(G,c)$ such that $T(G,c)|_{\alpha_{J_{i-1}}}=T(G,c)|_{\beta_{J_{i-1}}}$. Hence, there are exactly $2^{|E|-2|V|+\#G_{E\setminus J_{i-1}}+\#G_{J_{i-1}}}$ accepting paths going through s. $\square$

Lemma 39. *The size of any* 1-NBP *computing a satisfiable* $\mathrm{T}(G,c)$ *is at least the minimal size of* OBDD *computing* $\mathrm{T}(G,c)$.

Proof. Consider a minimal read-once nondeterministic branching program D computing $\mathrm{T}(G,c)$. For every satisfying assignment of $\mathrm{T}(G,c)$ there exists a corresponding accepting path in D. Consider a set of accepting paths P that contains exactly one path for every satisfying assignment. Notice that in 1-NBP one assignment may correspond to multiple paths. For every node v of D except the 0-sink there is at least one path of P, since otherwise v can be joined with 0-sink and it will decrease the size of D.

For every node v of D we denote by $q(v)$ the number of paths from P passing through v. For every $p \in P$, we consider the value $\gamma(p) = \sum_{v \in p} \frac{1}{q(v)}$, where summation goes over all vertices of p.

Notice that $\sum_{p \in P} \gamma(p) = |D| - 1$, since every node v except the 0-sink was calculated $q(v)$ times with weight $\frac{1}{q(v)}$. Hence, $|D| - 1 \geq |P| \min\limits_{p \in P} \gamma(p)$. Let $\min\limits_{p \in P} \gamma(p)$ be achieved on a path $p^* \in P$. Let π be a permutation corresponding to the order of the edges in p^* in the direction from the source to the 1-sink. Let D' be a minimal $\pi-$OBDD computing $T(G,c)$. For any node v of D' we denote by $q'(v)$ the number of accepting paths passing through v. For any path p in D' we define $\gamma'(p) = \sum_{v \in p} \frac{1}{q'(v)}$. By Lemma 38, the number of accepting paths passing through a vertex D' on a given level depends only on the number of this level and does not depend on particular node from it. Hence, $\gamma'(p)$ does not depend on p. Let P' be a set of accepting paths in D', then $|P| = |P'| = |A_{G,c}|$. Thus, $|D'| - 1 = \sum_{p \in P'} \gamma'(p) = |P'|\gamma(p')$, where p' is the path in D' corresponding to the path p^* in D. Consider the path p^* in D and enumerate all nodes labeled with variables: v_1 is labeled with $x_{e_{\pi(1)}}$, v_2 is labeled with $x_{e_{\pi(2)}}$, etc., $v_{|E|}$ is labeled with $x_{e_{\pi(|E|)}}$. Similarly consider the path p' in D'; it contains the following nodes: u_1 labeled with $x_{e_{\pi(1)}}$, u_2 labeled with $x_{e_{\pi(2)}}$, and etc., $u_{|E|}$ labeled with $x_{e_{\pi(|E|)}}$. By Lemma 38 and Proposition 36, for all $i \in [|E|]$ the inequality $q(v_i) \leq q'(u_i)$ holds. Hence, $\gamma(p') \leq \gamma(p^*)$, and, thus, $|D| \geq |D'|$. □

Remark 40. *Notice that the proof of Lemma 39 in fact gives a polynomial-time algorithm that given a nondeterministic read-once branching program D computing* $\mathrm{T}(G,c)$ *produces an ordered binary decision diagram D' computing* $\mathrm{T}(G,c)$ *such that $|D'| \leq |D|$. Indeed, for any node v of D we can easily compute $q(v)$: we compute number of paths from the source of D to v and from v to the 1-sink by dynamic programming. Then we use dynamic programming again in order to find a minimum-weight path p^* from the source of D to the 1-sink. Thus, we construct the permutation π corresponding to the path p^*. Now we just need to show that we can build a minimal $\pi-$*OBDD *computing* $\mathrm{T}(G,c)$ *in time that is polynomial of its size. We build $\pi-$*OBDD *level by level. On each level we join any two nodes that compute the same function. To do this we match each node with a Tseitin formula computed in this node, and join two nodes if and only if their Tseitin formulas are equal. It is easy to see that all this work can be performed in time* $\mathrm{poly}(|D|)$.

4.2 The component width

Let π be a permutation of the set $[|E|]$. We define π-compw(G) as a maximum over all i from 0 to $|E|$ of the value $\mathrm{comp}_{J_i}(G)$, where $J_i = \{e_{\pi(1)}, \ldots, e_{\pi(i)}\}$. We define *the component width* of the graph G (compw(G)) as the minimum over all permutations π of the value π-compw(G).

Theorem 41. *The size of any* 1-NBP *computing a satisfiable* $\mathrm{T}(G,c)$ *is at least* $2^{\mathrm{compw}(G)}$.

Proof. By Lemma 39, the size of 1-NBP computing $\mathrm{T}(G,c)$ is at least the minimal size of OBDD computing $\mathrm{T}(G,c)$. Let D be a minimal OBDD computing $\mathrm{T}(G,c)$. Let π be a permutation corresponding to the order of variables in D. By Lemma 38, the size of D is at least $2+\sum_{i=0}^{|E|-1} 2^{\mathrm{comp}_{J_i}(G)} > \sum_{i=0}^{|E|} 2^{\mathrm{comp}_{J_i}(G)} \geq 2^{\pi\text{-compw}(G)} \geq 2^{\mathrm{compw}(G)}$, in the first inequality we use that $\mathrm{comp}_{J_{|E|}}(G)=0$. □

On the other hand we can easily show the following.

Proposition 42. *There exists an* OBDD *computing a satisfiable formula* $\mathrm{T}(G,c)$ *based on* $G(V,E)$ *of size at most* $|E|2^{\mathrm{compw}(G)}+2$.

Proof. Let π be a permutation of $[|E|]$ such that $\mathrm{compw}\, G = \pi\text{-compw}\, G$. Let D be a minimal $\pi-$OBDD computing a satisfiable Tseitin formula $T(G,c)$. By Lemma 38, every level of D consists of at most $2^{\mathrm{compw}(G)}$ nodes. There are $|E|$ levels and two sinks, hence the size of D is at most $|E|2^{\mathrm{compw}(G)}+2$. □

Recall that a graph H is a minor of a graph G if H can be obtained from G by the sequence of the following three operations: edge deletion, edge contraction and vertex deletion. The following lemma verifies that the component width is minor-monotone.

Lemma 43. *Let* $G'(V',E')$ *be a minor of a graph* $G(V,E)$*, then* $\mathrm{compw}(G') \leq \mathrm{compw}(G)$.

Proof. Let π be a permutation of $[|E|]$ such that $\mathrm{compw}\, G = \pi\text{-compw}\, G$. Let D be a minimal $\pi-$OBDD computing a satisfiable Tseitin formula $T(G,c)$. By Lemma 38, every level of D consists of at most $2^{\mathrm{compw}(G)}$ nodes.

It is sufficient to prove the lemma under the assumption that G' is obtained from G by one of the following three operations: edge deletion, edge contraction and vertex deletion. We consider these three operations separately.

Edge deletion. Let G' be obtained from G by the deletion of edge e. Assume that there exists a satisfying assignment of $\mathrm{T}(G,c)$ that substitutes value $a \in \{0,1\}$ to x_e. Consider a branching program D' that can be obtained from D as follows: we delete from D all nodes labeled with x_e. Consider a node s of D labeled with x_e and let us denote by s' the endpoint of the edge outgoing from s labeled with a. If s is the source of D, then the source of D' is s'. Otherwise, we take all edges incoming to s in D and redirect them to s' in D'. Note that D' is obtained from D by the deletion of all nodes from some level and redirection of several edges, hence every level of D' contains at most $2^{\mathrm{compw}(G)}$ nodes. On the other hand D' computes a satisfiable Tseitin formula $\mathrm{T}(G',c')$, thus by Lemma 38, D' has a level of size at least $2^{\mathrm{compw}(G')}$. Hence, $\mathrm{compw}(G') \leq \mathrm{compw}(G)$.

Edge contraction. Let G' be obtained from G by the contraction of an edge e connecting vertices u and v. Let c' be a charge function defined on the vertices of G' that coincides with c on all vertices except the new vertex $\{u,v\}$ and $c'(\{u,v\}) = c(u)+c(v)$. It is proved in the paper [15] (see Lemmas 4.1 and 4.2 in ECCC version of [15]) that if we change all nodes in an 1-NBP computing $\mathrm{T}(G,c)$ labeled with x_e to guessing nodes, we obtain an 1-NBP computing $\mathrm{T}(G',c')$. Let D be the minimal $\pi-$OBDD computing $T(G,c)$, and let a nondeterministic branching program D' be obtained from D by the changing of all nodes labeled with x_e by guessing nodes. This transformation does not change the order of all other variables on every path, hence D' is a nondeterministic OBDD computing $\mathrm{T}(G',c')$. Every level of D' contains at most $2^{\mathrm{compw}(G)}$ nodes. On the other hand, D' computes the satisfiable formula $\mathrm{T}(G',c')$, thus by Lemma 38, it has a level of size at least $2^{\mathrm{compw}(G')}$. Hence, $\mathrm{compw}(G') \leq \mathrm{compw}(G)$.

Vertex deletion. The deletion of a vertex v can be represented as consequential deletion of all edges incident to v and the deletion of isolated vertex. Notice that the deletion of an isolated vertex decreases $|V|, \#G_J, \#G_{E\setminus J}$ and $\#G$ by 1, hence this operation does not change $\mathrm{comp}_J(G)$. $\square$

Proposition 44 (Theorem 4.5 in ECCC version of [15]). *For any satisfiable Tseitin formula* $\mathrm{T}(G, c)$ *there is an* OBDD *computing it that has at most* $2^{\mathrm{pw}(G)+1}$ *nodes on every level.*

Corollary 45. *For any graph* G*,* $\mathrm{compw}(G) \leq \mathrm{pw}(G) + 1$*.*

Proof. Consider a satisfiable Tseitin formula $\mathrm{T}(G, c)$. By Proposition 44, there is an OBDD computing $\mathrm{T}(G, c)$ such that every level contains at most $2^{\mathrm{pw}(G)+1}$ nodes. By Lemma 38, it has a level of size at least $2^{\mathrm{compw}(G)}$. Thus, $\mathrm{compw}(G) \leq \mathrm{pw}(G) + 1$. $\square$

4.3 Component-width is at least half of treewidth

In this subsection we prove that the component width of a graph G is $\Omega(\mathrm{tw}(G))$.

Consider a graph $G(V, E)$ and let $J \subseteq E$, let us denote by $P_J(G)$ the set of vertices that are not isolated in both G_J and $G_{E\setminus J}$.

For a graph H we denote by $\widetilde{H}$ a graph that is obtained from H by the removal of all isolated vertices.

Lemma 46. *Let* $G(V, E)$ *be a connected graph with at least two vertices. For any* $J \subseteq E$*,*

$$\mathrm{comp}_J(G) = |P_J(G)| + 1 - \#\widetilde{G}_J - \#\widetilde{G}_{E\setminus J}.$$

Proof. Since G is connected, $\mathrm{comp}_J(G) = |V| + 1 - \#G_J - \#G_{E\setminus J}$. Note that $V \setminus P_J(G)$ is the set of vertices that are isolated either in G_J or in $G_{E\setminus J}$. Since G is connected and it has at least two vertices, there are no vertices that are isolated in both G_J and $G_{E\setminus J}$ simultaneously. Hence,

$$|V \setminus P_J(G)| = \left(\#G_J - \#\widetilde{G}_J\right) + \left(\#G_{E\setminus J} - \#\widetilde{G}_{E\setminus J}\right).$$

The statement of the lemma follows. $\square$

Recall that a vertex v of a graph H is a *cut vertex* if $\#H - v > \#H$. A graph H is *biconnected* if H is connected and H does not have cut vertices.

Lemma 47. *Let* $G(V, E)$ *be a biconnected graph and* $J \subseteq E$*. Then for any connected component* C *of* $\widetilde{G}_J$*,*

$$|C \cap P_J(G)| \geq 2.$$

Proof. If $J = \emptyset$ or $J = E$, then $\widetilde{G}_J$ is an empty graph and it has no connected components. It follows that we can assume that $J \neq \emptyset$ and $J \neq E$ and, thus, $\widetilde{G}_J$ is non-empty. Since $\widetilde{G}_J$ contains no isolated vertices, all connected components of $\widetilde{G}_J$ are of size at least 2.

Let C be a connected component of $\widetilde{G}_J$. If $C = V$, then consider an edge $e \in E \setminus J$. Both of its endpoints are not isolated in both G_J and $G_{E\setminus J}$, hence $|C \cap P_J(G)| = |P_J(G)| \geq 2$. We now assume that $C \neq V$.

Since G is connected, there is an edge $e \in E$ connecting C with $V \setminus C$. Since C is a connected component in G_J, $e \in E \setminus J$, hence the endpoint of e from C is in $P_J(G)$. Hence, $C \cap P_J(G) \neq \emptyset$. Suppose that $C \cap P_J(G)$ consists of exactly one vertex u. Then any edge connecting C and $V \setminus C$

has u as one of its endpoints, since otherwise its endpoint from C should be also in $C \cap P_J(G)$. Thus, there is no edge between $C \setminus \{u\}$ (that is nonempty since $|C| \geq 2$) and $V \setminus C$, hence $G - u$ is disconnected which contradicts the biconnectivity of G. Hence, $|C \cap P_J(G)| \geq 2$. $\square$

Lemma 48. *Let $G(V, E)$ be a connected graph without cut vertices and $J \subseteq E$ be a subset of its edges. If $S \subseteq P_J(G)$ is such that all vertices in S are from the same connected component of $\widetilde{G}_J$, then $\text{comp}_J(G) \geq \frac{1}{2} \cdot |S|$.*

Proof. By Lemma 46, $\text{comp}_J(G) = |P_J(G)| + 1 - \#\widetilde{G}_J - \#\widetilde{G}_{E \setminus J}$. By Lemma 47, every connected component of $\widetilde{G}_{E \setminus J}$ has at least two vertices in $P_J(G)$, thus, $\#\widetilde{G}_{E \setminus J} \leq \frac{|P_J(G)|}{2}$.

By the condition of the lemma, one of the connected components of $\widetilde{G}_J$, say C, has at least $|S|$ vertices in $P_J(G)$. By Lemma 47, each other connected component of $\widetilde{G}_J$, has at least two vertices in $P_J(G)$. Since each of them is disjoint with C, every connected component of $\widetilde{G}_J$ except C has at least two vertices in $P_J(G) \setminus S$. Hence, $\#\widetilde{G}_J \leq 1 + \frac{|P_J(G) \setminus S|}{2}$.

We finally get that

$$\text{comp}_J(G) = |P_J(G)| + 1 - \#\widetilde{G}_J - \#\widetilde{G}_{E \setminus J} \geq |P_J(G)| + 1 - \left(1 + \frac{|P_J(G) \setminus S|}{2}\right) - \frac{|P_J(G)|}{2} = \frac{|S|}{2}.$$

$\square$

Lemma 49. *Let $G(V, E)$ be a biconnected graph with m edges and π be a permutation of $[m]$. G admits a tree decomposition with the maximum bag size at most $2 \cdot \pi\text{-compw}(G) + 2$.*

Proof. We provide an explicit construction of a tree decomposition of G based on a given permutation π of $[m]$. We will use Lemma 48 to show that the maximum bag size of the constructed tree decomposition of G is at most $\pi\text{-compw}(G) \cdot 2 + 2$.

For $i \in [m]$ we denote $J_i = \{e_{\pi(1)}, e_{\pi(2)}, \ldots, e_{\pi(i)}\}$. Let us consider the sequence $P_{J_1}(G), P_{J_2}(G), \ldots, P_{J_m}(G)$. For each $i \in [m]$, we denote the connected components of $\widetilde{G}_{J_i}$ by $C_{i,1}, C_{i,2}, \ldots, C_{i,t_i}$, where $t_i = \#\widetilde{G}_{J_i}$. Let us denote $T'_{i,j} = C_{i,j} \cap P_{J_i}(G)$ for all $i \in [m], j \in [t_i]$.

We shall now ensure that both endpoints of $e_{\pi(i)}$ are contained in some set of this partition. Let $e_{\pi(i)}$ connect u and v. The vertices u and v are from the same connected component of $\widetilde{G}_{J_i}$, hence $u, v \in C_{i,k_i}$ for the unique $k_i \in [t_i]$. Let $T_{i,j} = T'_{i,j}$ for each $j \in [t_i] \setminus \{k_i\}$ and $T_{i,k_i} = T'_{i,k_i} \cup \{u, v\}$.

Claim 50. *For all $i \in [m]$, $T_{i,1}, T_{i,2}, \ldots, T_{i,t_i}$ is a partition of the set $P_{J_i}(G) \cup \{u, v\}$, where u and v are the endpoints of $e_{\pi(i)}$.*

Proof. Let us fix $i \in [m]$. Notice that $T_{i,j} \subseteq C_{i,j}$ for all $j \in [t_i]$. Since $C_{i,j}$ are disjoint for all $j \in [t_i]$, $T_{i,j}$ are also disjoint for $j \in [t_i]$. Let w be a vertex from $P_{J_i}(G) \cup \{u, v\}$. The graph $\widetilde{G}_{J_i}$ contains w. Hence, there exists $j \in [t_i]$ such that $w \in C_{i,j}$. Since $T_{i,j} = (P_{J_i}(G) \cup \{u, v\}) \cap C_{i,j}$, $w \in T_{i,j}$. $\square$

The constructed sets $T_{i,j}$ for $i \in [m]$ and $j \in [t_i]$ are the bags of our desired tree decomposition. Now we describe the edges between these bags. For each $i \in [m-1]$ and $j \in [t_i]$, we will introduce an edge between the bag $T_{i,j}$ and a bag $T_{i+1,p(i,j)}$ for some $p(i,j) \in [t_{i+1}]$. Thus, the resulting tree can be viewed as a tree rooted in the bag $T_{m,1}$ (note that $t_m = 1$), and for each $i \in [m]$ and $j \in [t_i]$, $T_{i,j}$ is a node of this tree that is at the distance $m - i$ from the root. In other words, $\{T_{m,1}\}, \{T_{m-1,1}, \ldots, T_{m-1,t_{m-1}}\}, \ldots, \{T_{1,1}, \ldots, T_{1,t_1}\}$ are the layers of this tree from the root to the leaves. An edge from $T_{i,j}$ to $T_{i+1,p(i,j)}$ is an edge going from a child node to its parent node in this rooted tree.

Now we define $p(i,j)$ for each $i \in [m-1]$ and $j \in [t_i]$. Recall that $T_{i,j} \subseteq C_{i,j}$, where $C_{i,j}$ is a connected component of $\widetilde{G}_{J_i}$.

There is the unique connected component $C_{i+1,j'}$ of $\widetilde{G}_{J_{i+1}}$ that contains $C_{i,j}$. Let $p(i,j) = j'$.

Let us verify that the bags $T_{i,j}$ with described edges between them form the tree decomposition of G (which we call $\mathcal{T}$).

Firstly, note that endpoints of each edge of G appear simultaneously in some bag of $\mathcal{T}$. Indeed, any edge of G can be represented as $e_{\pi(i)}$ for the unique $i \in [m]$ and endpoints of $e_{\pi(i)}$ are in T_{i,k_i} by the construction.

Secondly, let us prove that for all $v \in V$ the set of bags of $\mathcal{T}$ containing v forms a path in $\mathcal{T}$. Let ℓ be the minimal i such that $e_{\pi(i)}$ is incident to v and let r be the maximal such i.

Claim 51. *For each $i \in [m]$, $i \in [\ell, r]$ iff $v \in P_{J_i}(G) \cup \{u, w\}$, where u and w are the endpoints of $e_{\pi(i)}$.*

Proof. Assume that $i < \ell$ or $i > r$. Then $v \notin P_{J_i}(G)$, since v is isolated in G_{J_i} or $G_{E \setminus J_i}$. Also v is not an endpoint of $e_{\pi(i)}$ by the definition of ℓ and r. Thus, $v \notin P_{J_i}(G) \cup \{u, w\}$. Now, assume that $i \in [\ell, r]$. If $i = r$, then $v \in \{u, w\} \subseteq P_{J_i}(G) \cup \{u, w\}$. If $i \in [\ell, r-1]$, $v \in P_{J_i}(G)$ since J_i contains at least one edge incident to v but not all of them. □

By Claims 51 and 50, for all $i \in [\ell, r]$ there exists the unique $j_i \in [t_i]$ such that $v \in T_{i,j_i}$. In other words, v is contained in the bags $T_{\ell,j_\ell}, T_{\ell+1,j_{\ell+1}}, T_{r,j_r}$ and only in them. In order to show that these bags form a path, we have to verify that $p(i, j_i) = j_{i+1}$ for all $i \in [\ell, r-1]$. Indeed, $v \in T_{i,j_i} \subseteq C_{i,j_i}$. Analogously, $v \in C_{i+1,j_{i+1}}$, hence $C_{i,j_i} \subseteq C_{i+1,j_{i+1}}$ and, thus, $p(i, j_i) = j_{i+1}$. Hereby, $\mathcal{T}$ is a tree decomposition of G.

Now we estimate the size of the bags in $\mathcal{T}$. Consider some $i \in [m]$ and $j \in [t_i]$, $|T_{i,j}|$ is at most $|C_{i,j} \cap P_{J_i}(G)| + 2$. By Lemma 48 applied to $S = C_{i,j} \cap P_{J_i}(G)$ and $J = J_i$, we get that $2 \cdot \operatorname{comp}_{J_i}(G) \geq |C_{i,j} \cap P_{J_i}(G)|$. Hence, $|T_{i,j}| \leq 2 \cdot \operatorname{comp}_{J_i}(G) + 2 \leq 2 \cdot \pi\text{-compw}(G) + 2$. □

Notice that a tree decomposition constructed in Lemma 49 is a *special tree decomposition*, i.e. for every vertex the set of bags contained it forms a path.

Theorem 52. *For any graph G,* $\operatorname{compw}(G) \geq \frac{1}{2}(\operatorname{tw}(G) - 1)$.

Proof. Recall that a block of G is a maximal biconnected subgraph of G. Let $B_1, B_2, \ldots, B_k$ be the blocks of G.

The treewidth of a graph is equal to the maximum treewidth of its blocks [7], so $\operatorname{tw}(G) = \max_{i=1}^{k} \operatorname{tw}(B_i)$. Thus, $\operatorname{tw}(G) = \operatorname{tw}(B_j)$ for some $j \in [k]$. By Lemma 43, $\operatorname{compw}(G) \geq \operatorname{compw}(B_j)$. On the other hand, by Lemma 49, $\operatorname{tw}(B_j) \leq 2 \cdot \min_{\pi \in S_m} \pi\text{-compw}(B_j) + 1 = 2 \cdot \operatorname{compw}(B_j) + 1$. Finally obtain that $\operatorname{tw}(G) = \operatorname{tw}(B_j) \leq 2 \cdot \operatorname{compw}(B_j) + 1 \leq 2 \cdot \operatorname{compw}(G) + 1$.

□

4.4 Component width may be close to pathwidth

In Corollary 45 we notice that $\operatorname{compw}(G) \leq \operatorname{pw}(G) + 1$. It is well-known that $\operatorname{tw}(G) \leq \operatorname{pw}(G) \leq \mathcal{O}(\operatorname{tw}(G) \log n)$. A well-known example of graphs with logarithmic multiplicative gap between treewidth and pathwidth are complete binary trees. A complete binary tree of height h has $2^{h+1} - 1$

vertices, treewidth 1 and pathwidth $\lceil \frac{h}{2} \rceil$ [26]. It is easy to see that the component width of any tree equals zero. Indeed, if $G(V,E)$ is a tree, then for every $J \subseteq E$, $\#G_J = |V| - |J|$; hence $\mathrm{comp}_J(G) = |V| - \#G_J - \#G_{E \setminus J} + 1 = |V| - (|V| - |J|) - (|V| - |E| + |J|) + 1 = 0$.

In this section we give an example of family of constant-degree graphs G_n such that $\mathrm{compw}(G_n) = \Omega(\mathrm{tw}(G_n) \log n)$, where n is the number of vertices in G_n. Our construction uses the following plan:

1. We show that if a graph has the specific form (it can be represented by the strong product of some graph with a clique), the the component width is lower bounded by the pathwidth (Theorem 59).

2. We consider the strong product of a complete binary tree and a complete graph. We will see that such graph has almost all desired properties but it has unbounded degrees. In order to bound the degrees we use the result of [23] (see Proposition 61 for the statement).

Lemma 53. *Let graph $G(V,E)$ have m edges and π be a permutation of $[m]$. We use the notation $J_i = \{e_{\pi(1)}, \dots, e_{\pi(i)}\}$. Let u_i and v_i be the endpoints of $e_{\pi(i)}$. Then the sequence of sets $P_{J_1}(G) \cup \{u_1, v_1\}, P_{J_2}(G) \cup \{u_2, v_2\}, \dots, P_{J_m}(G) \cup \{u_m, v_m\}$ forms a path decomposition of G.*

Proof. By the construction, the endpoints of every edge appear simultaneously in $P_{J_i}(G) \cup \{u_i, v_i\}$ for some i. It is left to show that each vertex appears in a consecutive interval of bags. Consider an arbitrary vertex $v \in V$. Let ℓ be the minimum number such that $e_{\pi(\ell)}$ is incident to v, and r be the maximum such number. By the definition of $P_{J_i}(G)$, $v \in P_{J_i}(G)$ if and only if $i \geq \ell$ and $i < r$. If $v \in \{u_i, v_i\}$, then v is incident to $e_{\pi(i)}$, hence $\ell \leq i \leq r$. Also note that $v \in \{u_r, v_r\}$. Thus, $v \in P_{J_i}(G) \cup \{u_i, v_i\}$ if and only if $i \geq \ell$ and $i \leq r$. So we get that v appears in a consecutive interval of bags. □

Definition 54 ([25])**.** A *strong product* of graphs $G(V_G, E_G)$ and $H(V_H, E_H)$ is a graph $G \boxtimes H$ such that

- The set of vertices of $G \boxtimes H$ is $V_G \times V_H$
- There is an edge between (u,v) and (u',v') in $G \boxtimes H$ if and only if
  - $u = u'$ and $(v,v') \in E_H$ or
  - $v = v'$ and $(u,u') \in E_G$ or
  - $(u,u') \in E_G$ and $(v,v') \in E_H$.

Proposition 55. *For any graph G, $\mathrm{tw}(G \boxtimes K_k) + 1 \leq k(\mathrm{tw}(G) + 1)$ and $\mathrm{pw}(G \boxtimes K_k) + 1 \leq k(\mathrm{pw}(G) + 1)$, where K_k is the complete graph on the set of vertices $[k]$.*

Proof. Consider a tree decomposition of G of width $\mathrm{tw}(G)$ and replace each vertex v in each bag of the tree decomposition with all vertices in the set $\{v\} \times [k]$. It is easy to verify that as the result we obtain a tree decomposition of $G \boxtimes K_k$.

The proof for pathwidth and path decompositions is the same. □

Lemma 56. *Let $G(V,E)$ be a graph and $k \geq 1$ be an integer. Let $B_1, B_2, \dots, B_t$ be an arbitrary path decomposition of $G \boxtimes K_k$. There is an integer $i \in [t]$ and a set $S \subseteq V$ such that $|S| = \mathrm{pw}(G) + 1$ and $S \times [k] \subseteq B_i$.*

Proof. For each vertex (u, v) of the graph $(G \boxtimes K_k)$, we denote by $[\ell_{(u,v)}, r_{(u,v)}]$ the interval of indices i such that B_i contains (u, v).

We will now modify the path decomposition $B_1, B_2, \ldots, B_t$ in such a way that for each $u \in V$ and each $v, v' \in [k]$, vertices (u, v) and (u, v') appear in the same set of bags.

We will use several times the 1-dimensional Helly's theorem: given a set of intervals on a line if any two of them have a common point, then all intervals has a common point.

For any vertex $u \in V$, vertices of the set $\{u\} \times [k]$ induce a clique in $G \boxtimes K_k$, hence for every $v, v' \in [k]$ the intervals $[\ell_{(u,v)}, r_{(u,v)}]$ and $[\ell_{(u,v')}, r_{(u,v')}]$ have a common point. Thus, the intersection of all intervals $[\ell_{(u,v)}, r_{(u,v)}]$ for $v \in [k]$ is a nonempty interval $[L_u, R_u]$. We replace each of these k intervals with their intersection. In other words, for each $u \in V$ we remove all vertices in $\{u\} \times [k]$ from B_i for each $i \notin [L_u, R_u]$. Let us denote the resulting sequence of bags by $B'_1, B'_2, \ldots, B'_t$.

Let us verify that the endpoints of every edge of $G \boxtimes K_k$ appear in B'_i for some $i \in [t]$. Let (u', v') be a neighbor of (u, v). If $u = u'$, then both (u, v) and (u', v') are contained in some B'_i for some $i \in [L_u, R_u]$. We now assume that $u \neq u'$. Then for any $w, w' \in [k]$, (u, w) is a neighbor of (u', w'). Hence, for any $w, w' \in [k]$ the intervals $[\ell_{(u,w)}, r_{(u,w)}]$ and $[\ell_{(u',w')}, r_{(u',w')}]$ have a common point. Thus, for all $w \in [k]$ the interval $[\ell_{(u,w)}, r_{(u,w)}]$ has a common point with $[L_{u'}, R_{u'}]$. Finally, $[L_u, R_u]$ has a common point i with $[L_{u'}, R_{u'}]$. We get that $(u, v), (u', v') \in B'_i$. Thus, we have verified that $B'_1, B'_2, \ldots, B'_t$ is a path decomposition of $G \boxtimes K_k$.

Note that for each $i \in [t]$, B'_i can be represented as $B''_i \times [k]$ for the unique $B''_i \subseteq V$. It is easy to see that $B''_1, B''_2, \ldots, B''_t$ is a path decomposition of G. Then there exists $i \in [t]$ such that $|B''_i| \geq \mathrm{pw}(G) + 1$. Since $B'_i \subseteq B_i$, we get that $B''_i \times [k] \subseteq B_i$. This completes the proof. ☐

Corollary 57. *For any graph G and any integer $k \geq 1$, $\mathrm{pw}(G \boxtimes K_k) + 1 = k(\mathrm{pw}(G) + 1)$.*

The following lemma extends Lemma 48.

Lemma 58. *Let $G(V, E)$ be a biconnected graph and $J \subseteq E$ be a subset of its edges. If $C_1, C_2, \ldots, C_k$ are disjoint subsets of $P_J(G)$ such that for each $j \in [k]$, C_j consists of vertices belonging to the same connected component of either $\widetilde{G}_J$ or $\widetilde{G}_{E \setminus J}$, then*

$$\mathrm{comp}_J(G) \geq \frac{1}{2} \cdot \sum_{j=1}^{k} |C_j| - k + 1.$$

Proof. By Lemma 46, $\mathrm{comp}_J(G) = |P_J(G)| + 1 - \#\widetilde{G}_J - \#\widetilde{G}_{E \setminus J}$.

Without loss of generality, assume that for each $i \in [t]$, vertices in C_i belong to the same connected component of $\widetilde{G}_J$, and for each $i \in \{t+1, t+2 \ldots, k\}$ vertices in C_i belong to the same connected component of $\widetilde{G}_{E \setminus J}$. Let $S_1 = \bigcup_{i=1}^{t} C_i$ and $S_2 = \bigcup_{i=t+1}^{k} C_i$. Since $C_1, \ldots, C_k$ are pairwise disjoint, $|S_1| + |S_2| = \sum_{i=1}^{k} |C_i|$.

By Lemma 47, every connected component of $\widetilde{G}_J$ has at least two vertices in $P_J(G)$. Every connected component of $\widetilde{G}_J$ either contains C_i for some $i \in [t]$ or has at least two common vertices with $P_J(G) \setminus S_1$, hence $\#\widetilde{G}_J \leq t + \frac{|P_J(G) \setminus S_1|}{2}$. Analogously, $\#\widetilde{G}_{E \setminus J} \leq (k - t) + \frac{|P_J(G) \setminus S_2|}{2}$.

We finally get that

$$\mathrm{comp}_J(G) = |P_J(G)|+1-\#\widetilde{G}_J-\#\widetilde{G}_{E\setminus J} \geq |P_J(G)|+1-\left(t+\frac{|P_J(G)\setminus S_1|}{2}\right)-\left((k-t)+\frac{|P_J(G)\setminus S_2|}{2}\right) = \frac{|S_1|}{2}+\frac{|S_2|}{2}+1-k = \frac{1}{2}\sum_{i=1}^{k}|C_i|-k+1.$$

□

Theorem 59. *For any connected graph H and any integer $k \geq 3$,*

$$\mathrm{compw}(H \boxtimes K_k) \geq \frac{1}{2}\mathrm{pw}(H \boxtimes K_k) - \mathrm{pw}(H).$$

Proof. Let $G = H \boxtimes K_k$. It is sufficient to prove that π-compw$(G) \geq \frac{1}{2}\mathrm{pw}(G) - \mathrm{pw}(H)$ for each permutation π of the edges of G.

Take an arbitrary permutation $\pi \in S_m$, where m is the number of edges of G. Denote the edges of G by $e_1, e_2, \ldots, e_m$. Let $J_i = \{e_{\pi(1)}, \ldots, e_{\pi(i)}\}$. By Lemma 53, $P_{J_1}(G) \cup \{u_1, v_1\}, P_{J_2}(G) \cup \{u_2, v_2\}, \ldots, P_{J_m}(G) \cup \{u_m, v_m\}$ is a path decomposition of G, where u_i, v_i are the endpoints of $e_{\pi(i)}$. By Lemma 56, there exists an integer $i \in [m]$ and a set $S \subseteq V_H$, such that $|S| = \mathrm{pw}(H) + 1$ and $S \times [k] \subseteq P_{J_i}(G) \cup \{u_i, v_i\}$.

We claim that either $P_{J_i}(G)$ or $P_{J_{i-1}}(G)$ contains all vertices of $S \times [k]$ except, perhaps, one (note that $J_0 = \emptyset$, hence $P_{J_0}(G) = \emptyset$). Indeed, if $P_{J_i}(G)$ contains at least one endpoint of $e_{\pi(i)}$, the claim holds true. Otherwise, no endpoint of $e_{\pi(i)}$ is contained in the set $P_{J_i}(G)$, hence $e_{\pi(i)}$ is the last edge incident to both u_i and v_i according to the permutation π. Since G is a graph of minimum degree at least $k-1 \geq 2$, both these endpoints should be contained in $P_{J_{i-1}}(G) = P_{J_i}(G) \cup \{u_i, v_i\}$.

Thus, for some $t \in \{i-1, i\}$, $P_{J_t}(G)$ contains at least $|S| \cdot k - 1 = (\mathrm{pw}(H)+1) \cdot k - 1$ vertices of $S \times [k]$. For each vertex $v \in S$, we denote

$$C_v := (\{v\} \times [k]) \cap P_{J_t}(G).$$

Note that for each $v \in S$, $|C_v| = k$, except for, perhaps, some $u \in S$ with $|C_u| = k-1$. Also, for each $v \in S$, C_v induces a complete graph in G. It is easy to see that the edge set of a complete graph cannot be partitioned in two edge sets each forming a disconnected graph. Thus, all vertices in C_v belong to the same connected component at least in one of $\widetilde{G}_{J_t}$ and $\widetilde{G}_{E\setminus J_t}$.

It is easy to see that G is biconnected graph, so we may apply Lemma 58 to G and the sets C_v.

$$\pi\text{-compw}(G) \overset{\text{(Lemma 58)}}{\geq} \frac{1}{2}\sum_{v\in S}|C_v| - |S| + 1 \geq \frac{1}{2}(k\cdot(\mathrm{pw}(H)+1)-1) - (\mathrm{pw}(H)+1) + 1 = \frac{k}{2}(\mathrm{pw}(H)+1) - \frac{1}{2} - \mathrm{pw}(H) \overset{\text{(Cor. 57)}}{=} \frac{1}{2}(\mathrm{pw}(G)+1) - \frac{1}{2} - \mathrm{pw}(H) = \frac{1}{2}\mathrm{pw}(G) - \mathrm{pw}(H).$$

□

Corollary 60. *For any connected graph H and any integer $k \geq 3$,*

$$\operatorname{compw}(H \boxtimes K_k) \geq \left(\frac{1}{2} - \frac{1}{k}\right) \cdot \operatorname{pw}(H \boxtimes K_k) + \frac{k-1}{k}.$$

Proof. Follows from Corollary 57 and Theorem 59. □

In order to implement the second part of the plan we require the following useful result.

Proposition 61 (Theorem 3.1 in [23])**.** *For any connected graph $H(V_H, E_H)$, there exists a graph $G(V_G, E_G)$ of maximum degree three, such that*

- *H is a minor of G;*
- *$|V_G| \leq 2|E_H|$;*
- *$\operatorname{tw}(G) \leq \operatorname{tw}(H) + 1$.*

Lemma 62. *For any integer $k \geq 3$ and any odd integer $h \geq 1$, there exists an n-vertex graph G of maximum degree three, such that*

- *$k(2^{h+1} - 1) \leq n < 4k^2(2^{h+1} - 1)$;*
- *$\operatorname{tw}(G) \leq 2k$;*
- *$\operatorname{pw}(G) \geq \frac{k}{2} \cdot \log \frac{n}{k^2} - 1$;*
- *$\operatorname{compw}(G) \geq \frac{k-2}{4} \cdot \log \frac{n}{k^2}$.*

Proof. Let $h \geq 1$ be an arbitrary odd integer and let $T(V_T, E_T)$ be a complete binary tree of height h. Thus, $|V_T| = 2^{h+1} - 1$ and $|E_T| = 2^{h+1} - 2$, $\operatorname{tw}(T) = 1$ and $\operatorname{pw}(T) = \lceil \frac{h}{2} \rceil = \frac{h+1}{2}$.

Let $H(V_H, E_H) = T \boxtimes K_k$. Note that $|V_H| = k \cdot \left(2^{h+1} - 1\right)$, $|E_H| = k^2|E_T| + \binom{k}{2}|V_T| = k^2(2^{h+1} - 2) + \binom{k}{2}(2^{h+1} - 1) < 2k \cdot |V_H|$, and by Proposition 55, $\operatorname{tw}(H) \leq (\operatorname{tw}(T) + 1) \cdot k - 1 = 2k - 1$. By Corollary 57,

$$\operatorname{pw}(H) = (\operatorname{pw}(T) + 1) \cdot k - 1 = \left(\frac{h+1}{2} + 1\right) \cdot k - 1 \geq \log\left(\frac{|V_H|}{k}\right) \cdot \frac{k}{2} + k - 1.$$

Now apply Proposition 61 to H and obtain graph G of maximum degree three, such that $\operatorname{tw}(G) \leq \operatorname{tw}(H) + 1 \leq 2k$. Let $n = |V_G|$. Then $n \leq 2|E_H| < 4k \cdot |V_H|$. Since H is a minor of G and it is well-known that pathwidth may only decrease when taking minors, and $|V_H| > \frac{n}{4k}$,

$$\operatorname{pw}(G) \geq \operatorname{pw}(H) > \log\left(\frac{n}{4k^2}\right) \cdot \frac{k}{2} + k - 1 = \frac{k}{2} \cdot \log \frac{n}{k^2} - \frac{\log 4}{2} k + k - 1 = \frac{k}{2} \cdot \log \frac{n}{k^2} - 1.$$

Since $H = T \boxtimes K_k$, by Corollary 60, $\operatorname{compw}(H) \geq \frac{k-2}{2k}\operatorname{pw}(H) + \frac{k-1}{k}$. Also, by Lemma 43, $\operatorname{compw}(G) \geq \operatorname{compw}(H)$. Then

$$\operatorname{compw}(G) \geq \frac{k-2}{2k}\operatorname{pw}(H) + \frac{k-1}{k} > \frac{k-2}{2k}\left(\frac{k}{2} \cdot \log \frac{n}{k^2} - 1\right) + \frac{k-1}{k} > \frac{k-2}{4} \cdot \log \frac{n}{k^2}.$$

□

Model \ Graph		$n \times n$ grid	n-vertex expander	Constant-degree n-vertex graph
Tree-like resolution	**LB**	$2^{\Omega(n)}$ [5]	$2^{\Omega(n)}$ [5] $\boxed{?}$	$2^{\Omega(\mathrm{tw}(G))}$ [5] ★
	UB	$2^{\mathcal{O}(n)}$ (folklore)	$2^{\mathcal{O}(n \log n)}$ [21, 3] $\boxed{?}$	$2^{\mathcal{O}(\mathrm{tw}(G) \log n)}$ [21, 3] ★
Regular resolution	**LB**	$n^{\omega(1)}$ [27]; $2^{\Omega(n)}$ [11]	$2^{\Omega(n)}$ [5]	$2^{\mathrm{tw}^{\Omega(1)}(G)}$ [12]; $2^{\Omega(\mathrm{tw}(G)/\log n)}$ $\boxed{?}$
	UB	$2^{\mathcal{O}(n)}$ (folklore)	$2^{\mathcal{O}(n)}$ [1]	$2^{\mathcal{O}(\mathrm{tw}(G)))}$ [1] ★
General resolution	**LB**	$2^{\Omega(n)}$ [11]	$2^{\Omega(n)}$ [5]	$2^{\mathrm{tw}^{\Omega(1)}(G)}$ [12] $\boxed{?}$
	UB	$2^{\mathcal{O}(n)}$ (folklore)	$2^{\mathcal{O}(n)}$ [1]	$2^{\mathcal{O}(\mathrm{tw}(G)))}$ [1] ★
1-BP computing a satisfiable $\mathrm{T}(G, c)$	**LB**	$2^{\Omega(n)}$ [15]	$2^{\Omega(n)}$ [14]	$2^{\mathrm{tw}^{\Omega(1)}(G)}$ [15]; $2^{\Omega(\mathrm{tw}(G))}$ ★
	UB	$2^{\mathcal{O}(n)}$ [15]	$2^{\mathcal{O}(n)}$ [15]	$2^{\mathcal{O}(\mathrm{tw}(G) \log n)}$ [15] ★

Figure 3: **LB** stands for lower bound, **UB** stands for upper bound; our results are highlighted with blue; if upper and lower bounds do not match, we label a bound by ★ if this bound can be achieved and we label a bound by $\boxed{?}$ if we do not know, whether this bound can be achieved or not.

Theorem 63. *There exists a family of constant-degree graphs G_m such that G_m has n vertices, where $n = \Omega(m^3)$ and $n = \mathcal{O}(m^4)$, $\mathrm{tw}(G_m) = \Theta(m)$, $\mathrm{pw}(G_m) = \Theta(m \log m)$ and $\mathrm{compw}(G_m) = \Theta(m \log m)$.*

Proof. Lemma 62 for $k = m$ and $h = 2\lfloor \log m \rfloor + 1$ yields a graph G_m with n vertices such that $n = \mathcal{O}(m^4)$ and $n = \Omega(m^3)$ with $\mathrm{tw}(G_m) = \mathcal{O}(m)$, $\mathrm{compw}(G_m) = \Omega(m \log m)$ and $\mathrm{pw}(G_m) = \Omega(m \log m)$. Since $\mathrm{pw}(G_m) \geq \Omega(m \log m)$, $\mathrm{tw}(G_m) = \Omega\left(\mathrm{pw}(G_m)/\log n\right) = \Omega(m)$. On the other hand, since $\mathrm{tw}(G_m) = \mathcal{O}(m)$, $\mathrm{pw}(G_m) = \mathcal{O}(\mathrm{tw}(G_m) \log m) = \mathcal{O}(m \log m)$. By Corollary 45, $\mathrm{compw}(G_m) = \Omega(\mathrm{tw}(G_m)) = \Omega(m \log m)$. □

Corollary 64. *Let S be the size of the smallest* 1-BP *computing* $\mathrm{SearchVertex}(G_m, c'_m)$*. Then size of any* 1-BP *computing a satisfiable* $\mathrm{T}(G_m, c_m)$ *is at least* $S^{\Omega(\log m)}$.

Proof. By Theorem 63, $\mathrm{tw}(G_m) = \mathcal{O}(m)$. By Theorem 5, $w(\mathrm{T}(G_m, c'_m)) = \Theta(\mathrm{tw}(L(G_m)))$. By Theorem 6, $\mathcal{O}(\mathrm{tw}(L(G_m))) = \mathcal{O}(\mathrm{tw}(G_m)\Delta(G_m)) = \mathcal{O}(\mathrm{tw}(G_m)) = \mathcal{O}(m)$. Thus, by Theorem 3, there exists a regular resolution refutation of $\mathrm{T}(G_m, c'_m)$ of size $2^{\mathcal{O}(m)}$. Therefore, $S = 2^{\mathcal{O}(m)}$.

On the other hand by Theorem 41 the size of any 1-NBP computing $\mathrm{T}(G_m, c_m)$ is $2^{\Omega(\mathrm{compw}(G_m))} = 2^{\Omega(m \log m)} = S^{\Omega(\log m)}$. □

Corollary 65. *Size of any decision tree computing* $\mathrm{SearchVertex}(G_m, c'_m)$ *is at least* $2^{\Omega(\mathrm{tw}(G_m) \log m)}$.

Proof. By Theorem 41, the size of any 1-NBP computing $\mathrm{T}(G_m, c_m)$ is $2^{\Omega(\mathrm{compw}(G_m))} = 2^{\Omega(m \log m)}$. Suppose there exists a decision tree of size S computing $\mathrm{SearchVertex}(G_m, c'_m)$. Then by Theorem 29 there exists a 1-BP of size $S + 1$ computing $\mathrm{T}(G_m, c_m)$. Therefore, $S = 2^{\Omega(\mathrm{tw}(G_m) \log m)}$. □

5 Conclusion

Our results are illustrated on Figure 3.

Open questions:

- Is it possible to prove that $S_R(\mathrm{T}(G,c)) \geq 2^{\Omega(\mathrm{tw}(G))}$?
- Is it possible to prove a similar lower bound for unrestricted resolution?
- Is it possible to separate $\mathrm{SearchT}(G,c)$ and $\mathrm{SearchVertex}(G,c)$ for constant degree graphs?

Acknowledgments. The authors thank Ludmila Glinskih, Alexander Knop, Alexander Smal and Navid Talebanfard for fruitful discussions. Dmitry Itsykson was partially supported by Young Russian Mathematics award.

References

[1] Michael Alekhnovich and Alexander A. Razborov. Satisfiability, branch-width and tseitin tautologies. *Computational Complexity*, 20(4):649–678, 2011.

[2] Albert Atserias. On digraph coloring problems and treewidth duality. *European Journal of Combinatorics*, 29(4):796 – 820, 2008. Homomorphisms: Structure and Highlights.

[3] Paul Beame, Chris Beck, and Russell Impagliazzo. Time-space trade-offs in resolution: Superpolynomial lower bounds for superlinear space. *SIAM J. Comput.*, 45(4):1612–1645, 2016.

[4] Eli Ben-Sasson. Size space tradeoffs for resolution. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 457–464, New York, NY, USA, 2002. ACM.

[5] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *J. ACM*, 48(2):149–169, 2001.

[6] Dan Bienstock. On embedding graphs in trees. *Journal of Combinatorial Theory, Series B*, 49(1):103 – 136, 1990.

[7] Hans L Bodlaender and Arie MCA Koster. Safe separators for treewidth. *Discrete Mathematics*, 306(3):337–350, 2006.

[8] Sam Buss, Dmitry Itsykson, Alexander Knop, and Dmitry Sokolov. Reordering Rule Makes OBDD Proof Systems Stronger. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference (CCC 2018)*, volume 102 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:24, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[9] Samuel R. Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes (abstract). In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity, Atlanta, Georgia, USA, May 4-6, 1999*, page 5, 1999.

[10] Julia Chuzhoy and Zihan Tan. Towards tight(er) bounds for the excluded grid theorem. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 1445–1464, Philadelphia, PA, USA, 2019. Society for Industrial and Applied Mathematics.

[11] Stefan S. Dantchev and Søren Riis. Tree resolution proofs of the weak pigeon-hole principle. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, Illinois, USA, June 18-21, 2001*, pages 69–75. IEEE Computer Society, 2001.

[12] Nicola Galesi, Dmitry Itsykson, Artur Riazanov, and Anastasia Sofronova. Bounded-depth frege complexity of tseitin formulas for all graphs. In *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, pages 49:1–49:15, 2019.

[13] Nicola Galesi, Navid Talebanfard, and Jacobo Torán. Cops-robber games and the resolution of tseitin formulas. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, volume 10929 of *Lecture Notes in Computer Science*, pages 311–326. Springer, 2018. Full version is available as ECCC technical report TR18-170.

[14] L. Glinskih and D. Itsykson. Satisfiable Tseitin formulas are hard for nondeterministic read-once branching programs. In *MFCS-2017*, pages 26:1–26:12, 2017.

[15] Ludmila Glinskih and Dmitry Itsykson. On Tseitin formulas, read-once branching programs and treewidth. In *Computer Science - Theory and Applications - 14th International Computer Science Symposium in Russia, CSR 2019, Novosibirsk, Russia, July 1-5, 2019, Proceedings*, pages 143–155, 2019. Full version is available as ECCC technical report TR19-20.

[16] Dima Grigoriev. Linear lower bound on degrees of Positivstellensatz calculus proofs for the parity. *Theor. Comput. Sci.*, 259(1-2):613–622, 2001.

[17] Daniel J. Harvey and David R. Wood. The treewidth of line graphs. *Journal of Combinatorial Theory, Series B*, 132:157 – 179, 2018.

[18] Johan Håstad. On small-depth frege proofs for tseitin for grids. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 97–108. IEEE Computer Society, 2017.

[19] D.M. Itsykson and A.A. Kojevnikov. Lower bounds of static Lovasz-Schrijver calculus proofs for Tseitin tautologies. *Zapiski Nauchnykh Seminarov POMI*, 340:10–32, 2006.

[20] Dmitry Itsykson, Alexander Knop, Andrey Romashchenko, and Dmitry Sokolov. On OBDD-Based Algorithms and Proof Systems That Dynamically Change Order of Variables. In Heribert Vollmer and Brigitte Vallee, editors, *34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*, volume 66 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[21] Dmitry Itsykson and Vsevolod Oparin. Graph expansion, tseitin formulas and resolution proofs for CSP. In Andrei A. Bulatov and Arseny M. Shur, editors, *Computer Science - Theory and Applications - 8th International Computer Science Symposium in Russia, CSR 2013, Ekaterinburg, Russia, June 25-29, 2013. Proceedings*, volume 7913 of *Lecture Notes in Computer Science*, pages 162–173. Springer, 2013.

[22] László Lovász, Moni Naor, Ilan Newman, and Avi Wigderson. Search Problems in the Decision Tree Model. *SIAM J. Discrete Math.*, 8(1):119–132, 1995.

[23] Igor L Markov and Yaoyun Shi. Constant-Degree Graph Expansions that Preserve Treewidth. *Algorithmica*, 59(4):461–470, 2011.

[24] Toniann Pitassi, Benjamin Rossman, Rocco A. Servedio, and Li-Yang Tan. Poly-logarithmic frege depth lower bounds via an expander switching lemma. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 644–657. ACM, 2016.

[25] Gert Sabidussi. Graph multiplication. *Mathematische Zeitschrift*, 72(1):446–457, 1959.

[26] Petra Scheffler. Optimal embedding of a tree into an interval graph in linear time. In *Annals of Discrete Mathematics*, volume 51, pages 287–291. Elsevier, 1992.

[27] G.S. Tseitin. On the complexity of derivation in the propositional calculus. In *Studies in Constructive Mathematics and Mathematical Logic Part II*. A. O. Slisenko, editor, a968.

[28] A. Urquhart. Hard examples for resolution. *JACM*, 34(1):209–219, 1987.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il