Theory Comput Syst (2018) 62:136–161
https://doi.org/10.1007/s00224-017-9814-5

Size-Treewidth Tradeoffs for Circuits Computing the Element Distinctness Function

Mateus de Oliveira Oliveira[1]

Published online: 5 October 2017
© Springer Science+Business Media, LLC 2017

Abstract In this work we study the relationship between size and treewidth of circuits computing variants of the element distinctness function. First, we show that for each n, any circuit of treewidth t computing the element distinctness function $\delta_n : \{0,1\}^n \to \{0,1\}$ must have size at least $\Omega(\frac{n^2}{2^{O(t)} \log n})$. This result provides a non-trivial generalization of a super-linear lower bound for the size of Boolean formulas (treewidth 1) due to Nečiporuk. Subsequently, we turn our attention to read-once circuits, which are circuits where each variable labels at most one input vertex. For each n, we show that any read-once circuit of treewidth t and size s computing a variant $\tau_n : \{0,1\}^n \to \{0,1\}$ of the element distinctness function must satisfy the inequality $t \cdot \log s \geq \Omega(\frac{n}{\log n})$. Using this inequality in conjunction with known results in structural graph theory, we show that for each fixed graph H, read-once circuits computing τ_n which exclude H as a minor must have size at least $\Omega(n^2/\log^4 n)$. For certain well studied functions, such as the *triangle-freeness* function, this last lower bound can be improved to $\Omega(n^2/\log^2 n)$.

Keywords Circuit complexity · Super-linear lower bounds · Element distinctness

This is an extended version of a paper that appeared at STACS 2016 [8].

The author is currently supported by the Bergen Research Foundation. This work was partially done while the author was a postdoctoral researcher at the Czech Academy of Sciences, supported by the European Research Council (grant agreement 339691).

This article is part of the Topical Collection on *Theoretical Aspects of Computer Science*

✉ Mateus de Oliveira Oliveira
mateus.oliveira@uib.no

[1] University of Bergen, Bergen, Norway

1 Introduction

The problem of explicitly defining a function in NP which requires super-linear circuit size has proven to be notoriously hard. Currently, the best known lower bound for a function in NP is of the order[1] of $3n - o(1)$ for circuits with arbitrary fan-in-2 gates [4, 9], and of the order of $5n - o(1)$ for circuits with gates from the binary De-Morgan basis [20, 23]. In the particular case of Boolean formulas, Nečiporuk proved an $\Omega(n^2/\log n)$ lower bound for the size of Boolean formulas over the full binary basis computing the n-bit element distinctness function [27]. Intuitively, the element distinctness function $\delta_n : \{0,1\}^n \to \{0,1\}$ takes as input a sequence of m numbers $s_1, s_2, ..., s_m \in \{1, ..., m^2\}$ encoded as binary strings with $2 \log m$ bits, and returns 1 if and only if all numbers in this sequence are distinct. Remarkably, Nečiporuk's lower bound has resisted improvements during the last four decades, and remains the strongest known lower bound for the size of formulas over the full binary basis. In the restricted setting of formulas over the De-Morgan basis, a size lower bound of $n^{3-o(1)}$ was obtained by Håstad [17] using different techniques.

In this work, we consider the problem of proving circuit size lower bounds for circuits of low treewidth. During the past decade a considerable amount of research has been devoted to the study of the computational power and the combinatorial properties of circuits parameterized by treewidth [1, 2, 6, 13, 14, 18, 21]. In our first result we generalize Nečiporuk's lower bound to the context of circuits of low treewidth.

Theorem 1 *Let $\mathcal{C}$ be a circuit of treewidth t computing the element distinctness function δ_n. Then $\mathcal{C}$ has size $\Omega(\frac{n^2}{2^{O(t)} \log n})$.*

Here the size of a circuit $\mathcal{C}$ is defined as its wire-complexity, i.e., the total number of edges in $\mathcal{C}$. Therefore, our lower bound holds for circuits containing unbounded fan-in AND and OR gates, and more generally, unbounded fan-in associatively constructible gates, which we will define in Section 2. Theorem 1 generalizes Nečiporuk's non-linear lower bound, from the context of Boolean formulas (that is to say, circuits of treewidth 1) to the context of circuits of low treewidth. In particular, our result implies an $\Omega(n^2/\log n)$ lower bound for the size of circuits whose underlying undirected graph belongs to several interesting classes, such as trees (treewidth at most 1), TTSP series-parallel graphs (treewidth at most 2), outer-planar graphs (treewidth at most 2), Halin graphs (treewidth at most 3), k-outerplanar graphs for fixed k (treewidth at most $O(k)$), etc. Additionally, Theorem 1 implies non-linear lower bounds even for circuits of treewidth $o(\log n)$.

It is worth comparing our result with another prominent restricted family of circuits for which no non-linear lower bound is known, namely, circuits whose underlying graph belongs to the class of Valiant Series-Parallel graphs [34]. We refer to

[1] Recently, this lower bound was improved to $(3 + 1/86)n - o(n)$ [12].

[7] for a clear definition of this class. It can be shown that the class of Valiant-series-parallel graphs strictly contains the class of TTSP-series-parallel graphs (which have treewidth 2). Nevertheless, Valiant-series-parallel graphs are incomparable with graphs of treewidth k, for $k \geq 3$. On the one hand, there are Valiant-series-parallel graphs of treewidth at least k for every $k \in \mathbb{N}$. For instance, the $k \times k$ grid-graph is Valiant-series-parallel but has treewidth k. On the other hand, it is easy to construct graphs of treewidth 3 which are not Valiant-series-parallel. Proving a non-linear lower bound for Valiant-series-parallel circuits remains a major open problem in circuit complexity [29, 31].

Next, we turn our attention to *read-once* circuits, which are circuits where each variable labels at most one input vertex. These circuits have also been known in the VLSI literature as *semilective* circuits [19]. Read-once circuits parameterized by treewidth have been studied by the SAT-solving and proof-complexity communities. Part of the interest in these circuits is due to the fact that the satisfiability problem for read-once circuits of size s and treewidth t can be solved in time $2^{O(t)} \cdot s^{O(1)}$ [1, 2, 6, 14]. Questions related to the design of optimal VLSI circuits have motivated the study of the complexity of *planar* read-once circuits computing explicit functions (i.e. functions in NP). Within this line of research, quadratic lower bounds have been obtained for the size of planar read-once circuits computing both multiple-output functions [24] and single-output functions [32]. We contrast these quadratic lower bounds with the fact that for multilective planar circuits, i.e., planar circuits in which variables can label arbitrarily many input gates, the best known lower bounds are of the order of $O(n \log n)$ for single-output functions and of the order of $O(n^{3/2})$ for multiple-output functions [33].

In this work we introduce the *symmetric non-deterministic state complexity* (symmetric-NSC) of a Boolean function, a complexity measure that is lower-bounded by the size of the smallest read-once oblivious branching program computing the function in question. We show that if $\mathcal{C}$ is a read-once circuit of size s and treewidth t computing a function $f_n : \{0,1\}^n \to \{0,1\}$ of symmetric-NSC $snsc(f_n)$, then $t \cdot \log s \geq \Omega(\log snsc(f_n))$. Using this tradeoff in conjunction with known results from structural graph theory, we show that for each fixed graph H, read-once H-minor-free[2] circuits computing f_n must have size at least $\Omega\left(\frac{\log^2 snsc(f_n)}{\log^2 n}\right)$. Subsequently, we introduce a variant $\tau_n : \{0,1\}^n \to \{0,1\}$ of the element distinctness function and show that its symmetric-NSC is lower bounded by $2^{\Omega(n/\log n)}$. From these results we have that read-once H-minor-free circuits computing τ_n require size $\Omega(n^2/\log^4 n)$. Near-quadratic lower bounds can also be obtained for the size of read-once H-minor-free circuits computing certain well studied functions, such as the *triangle-freeness* function Δ_n, and the *triangle-parity* function $\bigoplus \text{Clique}_{3,n}$. A result from [11] implies that the symmetric-NSC of these functions is lower-bounded by $2^{\Omega(n)}$. Therefore, read-once H-minor-free circuits computing both Δ_n and $\bigoplus \text{Clique}_{3,n}$ require size $\Omega(\frac{n^2}{\log^2 n})$.

[2] We say that a circuit $\mathcal{C}$ is H-minor-free if its underlying undirected graph excludes H as a minor.

2 Preliminaries

Let Σ be a finite set of symbols. A k-ary gate over Σ is a function $g : \Sigma^k \to \Sigma$ where $k \geq 1$. We say that a k-ary gate g is *associatively constructible* if either $k = 1$ or $k \geq 2$ and there exists an associative commutative operation $\oplus : \Sigma \times \Sigma \to \Sigma$ such that $g(x_1, ..., x_k) = x_1 \oplus ... \oplus x_k$. In this last case, we may alternatively say that g is a k-ary $\oplus$-gate. Unbounded fan-in AND and OR gates are clearly associatively constructible. An unbounded fan-in MOD_r gate can be simulated by an associatively constructible gate g that computes the sum of its inputs modulo r, together with a unary gate $g' : \Sigma \to \Sigma$ that returns 0 if this sum is congruent to $0 \mod r$, and which returns 1 otherwise.

An *associatively constructible circuit* with n inputs is a directed acyclic graph $\mathcal{C} = (V, E, \mathbf{g})$ where V is a set of vertices, E is a set of directed edges, and $\mathbf{g}$ is a function that labels each vertex $v \in V$ with a symbol from Σ, a gate over Σ, or a variable from $\{x_1, ..., x_n\}$. The function $\mathbf{g}$ must satisfy the following conditions.

1. If the in-degree of v is 0, then $\mathbf{g}(v)$ is either an element of Σ or a variable in $\{x_1, ..., x_n\}$.
2. If the in-degree of v is k, then $\mathbf{g}(v)$ is an associatively constructible k-ary gate over Σ.

Vertices of in-degree 0 are called *inputs*. An input is *initialized* if it is labeled with an element of Σ, and *uninitialized* if it is labeled with a variable. Vertices of out-degree 0 are called outputs. A formula is a circuit, with a unique output, whose underlying graph is a tree. We say that a circuit $\mathcal{C} = (V, E, \mathbf{g})$ is *read once* if no two input vertices are labeled with the same variable. Since our circuits may contain associatively constructible gates of unbounded fan-in and unbounded fan-out, we define the size $|\mathcal{C}|$ of a circuit $\mathcal{C}$ as the number of edges in $\mathcal{C}$. Below, we define the notion of tree-decomposition of a circuit.

Definition 1 (Tree-Decomposition and Treewidth) A tree-decomposition of a circuit $\mathcal{C} = (V, E, \mathbf{g})$ is a triple $T = (N, F, \beta)$ where (N, F) is a tree with set of nodes N and set of arcs F, and $\beta : N \to 2^V$ is a function that associates with each node $u \in N$ a subset $\beta(u)$ of vertices of V such that the following three conditions are satisfied.

1. $\bigcup_{u \in N} \beta(u) = V$.
2. For each edge $(v, v') \in E$ there is some $u \in N$ such that $\{v, v'\} \subseteq \beta(u)$.
3. For each vertex $v \in V$, the set of nodes $N(v) = \{u \in N \mid v \in \beta(u)\}$ induces a connected subtree of T.

The width of T is defined as $w(T) = \max_{u \in N} |\beta(u)| - 1$. The *treewidth* of a circuit $\mathcal{C}$, denoted $\boldsymbol{tw}(\mathcal{C})$, is the minimum width of a tree-decomposition of $\mathcal{C}$.

We note that the directions of the edges of a circuit play no role in the definition of tree-decomposition. In other words, only the underlying undirected graph of $\mathcal{C}$ is relevant when considering its treewidth.

Let $\mathcal{C} = (V, E, \mathbf{g})$ be an associatively constructible circuit. For each $v \in V$, we let $\mathit{indeg}(v) = |\{v' \mid (v', v) \in E\}|$ be the in-degree of v, and $\mathit{outdeg}(v) =$

$|\{v' \mid (v, v') \in E\}|$ be the out-degree of v. We let $\boldsymbol{\iota} : \Sigma \to \Sigma$ be the *identity gate*, which sends each element $a \in \Sigma$ to itself. We say that $\mathcal{C}$ is *normalized* if for every vertex $v \in V$, $outdeg(v) > 1$ implies that $\mathbf{g}(v) = \boldsymbol{\iota}$ and that $indeg(v) = 1$. We note that any associatively constructible circuit $\mathcal{C}$ of treewidth t can be transformed into a normalized circuit $\mathcal{C}'$ of size at most $2 \cdot |\mathcal{C}|$ and treewidth at most $2 \cdot t$ computing the same function as $\mathcal{C}$. Indeed, the transformation consists into splitting each vertex v with $outdeg(v) > 1$ into two vertices v_{in}, v_{out} connected by an edge (v_{in}, v_{out}), where v_{in} is labeled with the same gate as v, v_{out} is labeled with the identity gate $\boldsymbol{\iota}$, the in-neighbors of v in $\mathcal{C}$ are in-neighbors of v_{in} in $\mathcal{C}'$, and the out-neighbors of v in $\mathcal{C}$ are the out-neighbors of v_{out} in $\mathcal{C}'$.

Circuits with constant treewidth may have arbitrarily large maximum in-degree and maximum out-degree. Lemma 1 below states that, at the expense of a linear increase in size and treewidth, one can reduce the maximum in-degree and out-degree of a circuit $\mathcal{C}$ to a constant without changing the function computed by $\mathcal{C}$. A variant of Lemma 1 was originally proved in [25]. We include a self-contained proof of this lemma for completeness.

Lemma 1 (Markov-Shi [25]) *Let $\mathcal{C}$ be a associatively constructible circuit of treewidth t, computing a function $f : \Sigma^n \to \Sigma$. Then f can be computed by a normalized circuit $\mathcal{C}'$ of size $|\mathcal{C}'| = O(|\mathcal{C}|)$, maximum-degree at most 3, maximum in-degree at most 2, maximum out-degree at most 2, and treewidth at most $O(t)$. Additionally, for each variable x, the number of input vertices labeled with x in $\mathcal{C}$ is equal to the number of input vertices labeled with x in $\mathcal{C}'$.*

Proof First, we derive from $\mathcal{C}$ a normalized circuit $\hat{\mathcal{C}} = (V, E, \mathbf{g})$ of size at most $2 \cdot |\mathcal{C}|$ and treewidth at most $2 \cdot t$, which computes the same function as $\mathcal{C}$.

Let $T = (N, F, \beta)$ be a tree decomposition of $\hat{\mathcal{C}}$ of width at most $2 \cdot t$. We may assume that this decomposition satisfies the following additional properties.

1. (N, F) is a sub-cubic tree, i.e., each node has degree at most 3.
2. There is an injective function $\mu : E \to N$ such that $\mu(v, v')$ has degree at most 2 and for each $(v, v') \in E$, $\{v, v'\} \subseteq \beta(\mu(v, v'))$.

For each $v \in V$, let $N(v) = \{u \in N \mid v \in \beta(u)\}$ be the set of nodes of T whose associated bag contains the vertex v. Let $\rho : V \to N$ be an injective function satisfying the following properties.

1. $\rho(v) \in N(v)$.
2. If $outdeg(v) = 1$ then $\rho(v) = \mu(v, v')$ where v' is the unique vertex in V such that $(v, v') \in E$.
3. If $outdeg(v) > 1$ then $\rho(v) = \mu(v', v)$ where v' is the unique vertex in V such that $(v', v) \in E$.

Note that if $outdeg(v) = 0$ then $\rho(v)$ can be any node in $N(v)$. Note also that since $\hat{\mathcal{C}}$ is normalized, if $outdeg(v) \geq 1$ then $indeg(v) = 1$. Therefore, Condition 3 above is well defined. Since T is a tree decomposition, for each $v \in V$ the set $N(v)$ induces a connected subtree $T(v) = (N(v), F(v))$ of (N, F). Intuitively, for each $v \in V$, the function ρ chooses a root for the tree $T(v)$. Note that $\rho(v)$ and any other node

in $N(v)$ have at most two children. The leaves of the subtree $T(v)$ are the nodes in $N(v)$ at maximal depth with respect to the root $\rho(v)$. We say that a leaf u of $T(v)$ is useful if either $outdeg(v) \leq 1$ and there exists $(v, v') \in E$ such that $\mu(v, v') = u$, or if $outdeg(v) > 1$ and there exists $(v, v') \in E$ with $\mu(v, v') = u$. We say that the tree decomposition T is *reduced* with respect the pair (μ, ρ) if for every $v \in V$ all leaves in $N(v)$ are useful. We can assume without loss of generality that T is reduced, since if u is a non-useful leaf of $N(v)$, then we may safely remove v from the bag $\beta(u)$ and still have a valid tree decomposition of $\hat{\mathcal{C}}$ with at most the same width as T.

Now we are in a position to describe the construction of the circuit $\mathcal{C}' = (V', E', \mathbf{g}')$. Intuitively, each vertex v of the circuit $\mathcal{C}$ will correspond to a rooted binary tree $\tau(v)$ in $\mathcal{C}'$ which has the same shape as the subtree $T(v) = (N(v), F(v))$ of (N, F). If $outdeg(v) \leq 1$, then $\tau(v)$ is in-branching.[3] If $\mathbf{g}(v)$ is a k-ary $\oplus$ gate, then the nodes of $\tau(v)$ which have two children are labeled by $\mathbf{g}'$ with the binary $\oplus$ gate, while all other nodes are labeled with the identity gate $\boldsymbol{\iota}$. If v is labeled with a unary gate, then the root of $\tau(v)$ is labeled with $\mathbf{g}(v)$ and all other nodes of $\tau(v)$ are labeled with the identity gate $\boldsymbol{\iota}$. On the other hand, if $outdeg(v) > 1$, then $\tau(v)$ is out-branching. In this case, all nodes of $\tau(v)$ are labeled with the identity gate $\boldsymbol{\iota}$. Finally, the edge set E' is defined as follows. For each $(v, v') \in E$ with $outdeg(v) > 1$, we add an edge from a node of the out-branching tree $\tau(v)$ to a node of the in-branching tree $\tau(v')$. On the other hand for each $(v, v') \in E$ with $indeg(v) \leq 1$ we add an edge from the root of $\tau(v)$ to a node of $\tau(v')$. The precise way in which such edges are created is specified in the formal construction below.

$$
\begin{aligned}
V' &= \{[v, u] \mid v \in V,\ u \in N(v)\} \\
E' &= \{([v, u], [v', u]) \mid (v, v') \in E, \mu(v, v') = u\} \\
&\quad \cup \{([v, u], [v, u']) \mid v \in V, outdeg(v) = 1,\ u \text{ is a child of } u' \text{ in } N(v)\} \\
&\quad \cup \{([v, u], [v, u']) \mid v \in V, outdeg(v) \geq 1,\ u' \text{ is a child of } u \text{ in } N(v)\}
\end{aligned}
$$

Note that the first set corresponds to edges between distinct trees, while the second set corresponds to edges belonging to in-branching trees, and the third set corresponds to edges belonging to out-branching trees.

Finally, the gate map $\mathbf{g}'$ is formally defined as follows. For each $v \in V$ with $outdeg(v) > 1$, we set $\mathbf{g}'([v, u]) = \boldsymbol{\iota}$ for every $u \in N(v)$. On the other hand, for each $v \in V$ with $outdeg(v) \leq 1$, and each $u \in N(v)$ we proceed as follows.

1. If $\mathbf{g}(v)$ is a k-ary $\oplus$-gate for some associative commutative operation $\oplus$, then we set $\mathbf{g}'([v, u]) = \oplus$ if u has two children in $N(v)$, and set $\mathbf{g}'([v, u]) = \boldsymbol{\iota}$ if u has at most one child in $N(v)$.
2. If $\mathbf{g}(v)$ is a unary gate, then we set $\mathbf{g}'([v, u]) = \mathbf{g}(v)$ if u is the root of $N(v)$, and set $\mathbf{g}'([v, u]) = \boldsymbol{\iota}$ if u is not the root of $N(v)$. We note that in this case the subtree induced by $N(v)$ is a line, since $N(v)$ has a unique useful leave.
3. If v is an input of $\mathcal{C}$, then $N(v)$ has a single node u and we set $\mathbf{g}'([v, u]) = \mathbf{g}(v)$.

[3] An *in-branching* tree is a directed tree where all edges are oriented toward the root. An *out-branching* tree is a directed tree where all edges are oriented toward the leaves.

A tree decomposition $T' = (N, F, \beta')$ of $\mathcal{C}'$ can be obtained as follows.[4] For each $u \in N$, we set

$$\beta'(u) = \{[v, u] \mid v \in \beta(u)\} \cup \{[v, u'] \mid v \in \beta(u), u' \text{is a child or the parent of } u\}.$$

Since the tree (N, F) is ternary, the width of T' is at most 4 times the width of T. Note that the size of the circuit $\mathcal{C}'$ constructed above is of the order of $O(t \cdot |\mathcal{C}|)$. Nevertheless this size can be reduced to $O(|\mathcal{C}|)$, i.e., without the multiplicative dependence on t, by contracting maximal induced paths $v_1 v_2 .... v_r$ where all vertices are labeled with the identity gate into a single vertex labeled with the identity gate. Since contraction can only decrease the treewidth of a graph, the statement of the theorem is proved. □

Lemma 1 implies that to prove non-linear size lower bounds for associatively constructible circuits of constant treewidth, it is enough to prove non-linear lower bounds for circuits of constant treewidth and constant maximum fan-in and fan-out.

Below, we define the notion of *rooted carving decomposition* of a circuit, a variant of the notion of carving decomposition defined in [30]. If T is a rooted tree, we denote by $nodes(T)$ the set of all nodes of T, and by $leaves(T)$ the set of all leaves of T. For each node $u \in nodes(T)$, we let $T[u]$ denote the subtree of T rooted at u.

Definition 2 (Carving Decomposition) A *rooted carving decomposition* of a circuit $\mathcal{C} = (V, E, \mathbf{g})$ is a pair (T, γ) where T is a rooted binary tree and $\gamma : leaves(T) \to V$ is a bijection mapping each leaf $u \in leaves(T)$ to a single vertex $\gamma(u) \in V$.

Observe that the internal nodes of a carving decomposition T are unlabeled. Given a node $u \in nodes(T)$, we let $V(u) = \gamma(leaves(T[u])) = \{\gamma(v) \mid v \in leaves(T[u])\}$ be the image of the leaves of $T[u]$ under γ. For two distinct subsets V_1, V_2 of vertices of a circuit $\mathcal{C}$ we let $E(V_1, V_2)$ denote the set of edges in G with one endpoint in V_1 and another endpoint in V_2. The width $carw(T, \gamma)$ of the carving decomposition (T, γ) is defined as

$$\max\{|E(V(u), V \backslash V(u))| : u \in nodes(T)\}.$$

The carving width $carw(\mathcal{C})$ of a circuit $\mathcal{C}$ is the minimum width of a carving decomposition of $\mathcal{C}$. The following lemma relates carving width and treewidth of an associatively constructible circuit.

Lemma 2 *Let $\mathcal{C} = (V, E, \mathbf{g})$ be an associatively constructible circuit of treewidth t. There is a circuit $\mathcal{C}'$ of size $|\mathcal{C}'| = O(|\mathcal{C}|)$, maximum degree 3, and carving width $O(t)$ such that $\mathcal{C}$ and $\mathcal{C}'$ compute the same function. Additionally, for each variable x, the number of input vertices labeled with x in $\mathcal{C}$ is equal to the number of input vertices labeled with x in $\mathcal{C}'$.*

[4] The structure (N, F) of the decomposition remains the same. Only function assigning bags to nodes in N is updated.

Proof By Lemma 1, if $\mathcal{C}$ is an associatively constructible circuit of treewidth t, then one can construct a circuit $\mathcal{C}'$ of size $O(|\mathcal{C}|)$, maximum degree at most 3 and treewidth at most $O(t)$ which computes the same function as $\mathcal{C}$. Additionally, for each variable x, the number of input vertices labeled with x in $\mathcal{C}$ is equal to the number of input vertices labeled with x in $\mathcal{C}'$. Since any graph of treewidth w and maximum degree Δ has carving-width $O(\Delta \cdot w)$ (Lemma 4 of [26]), the carving width of $\mathcal{C}'$ is bounded by $O(t)$. □

3 Nečiporuk's Method

In this section, we briefly describe Nečiporuk's method for proving non-linear lower bounds on the size of Boolean formulas over the complete binary basis. For our purposes, it will be convenient to divide this method into three steps. Our first main result (Theorem 1) follows from a generalization of Step 1 given below. A complete proof of Nečiporuk's theorem can be found in [22].

Step 1 Let $X = \{x_1, ..., x_n\}$ be a set of variables, $f : \{0,1\}^X \to \{0,1\}$ be a Boolean function on X, and $Y \subseteq X$ be a subset of variables of X. We denote by $N_f(Y)$ the number of distinct functions that can be obtained by initializing all variables in $X \backslash Y$ with values in $\{0,1\}$. The first step in the proof of Nečiporuk's theorem consists in providing an upper bound for $N_f(Y)$. If f can be computed by a Boolean formula F, such an upper bound can be given in terms of the number of inputs of F labeled with variables in Y.

Proposition 1 *Let $f : \{0,1\}^X \to \{0,1\}$ be a function computable by a Boolean formula F. Let $Y \subseteq X$ be a subset of variables such that at most l inputs of F are labeled with variables in Y. Then $N_f(Y)$ is at most $2^{O(l)}$.*

We briefly sketch the proof of Proposition 1. Let $g : \{0,1\}^Y \to \{0,1\}$ be a function obtained from f by initializing all variables in $X \backslash Y$ according to an assignment $\alpha : X \backslash Y \to \{0,1\}$. Let F_g be the Boolean formula obtained from F by initializing inputs labeled with variables in $X \backslash Y$ according to α. Then, F_g computes g, and has at most l uninitialized inputs. Now, the formula F_g can be shrunk into a formula F'_g which still computes the function g, but which satisfy the following two properties.

1. F'_g has at most l inputs, all of which are labeled with variables in Y.
2. All internal nodes of F'_g have fan-in (precisely) 2.

These two properties imply that F'_g has at most $l-1$ internal nodes. Since there are 16 possible Boolean functions of fan-in 2, there are at most $16^{l-1} = 2^{O(l)}$ choices for g. Therefore, $N_f(Y)$ is at most $2^{O(l)}$.

Step 2 The second step consists in exhibiting an explicit Boolean function with many sub-functions. Intuitively, a function $f : \{0,1\}^X \to \{0,1\}$ has many sub-functions if the quantity $N_f(Y)$ is large for some subsets $Y \subseteq X$ of suitable size. Let $X = \{x_1, ..., x_n\}$ be a set of $n = 2m \log m$ distinct variables partitioned into m blocks

$Y_1, Y_2, ..., Y_m$, where each block Y_i has $2 \log m$ variables. The *element distinctness* function $\delta_n : \{0,1\}^X \to \{0,1\}$ is defined as follows for each assignment $s_1, s_2, ..., s_m$ of the blocks $Y_1, Y_2, ..., Y_m$ respectively.

$$\delta_n(s_1, s_2, ..., s_m) = \begin{cases} 1 & \text{if } s_i \neq s_j \text{ for } i \neq j, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

The following lemma states that the element distinctness function defined in (1) has many sub-functions.

Lemma 3 *(See [22], Section 6.5) Let $\delta_n : \{0,1\}^X \to \{0,1\}$ be the element distinctness function defined in (1), where $|X| = n$ and $X = Y_1 \dot\cup Y_2 \dot\cup ... \dot\cup Y_m$ with $|Y_i| = 2 \log m$. Then for each $i \in \{1, ..., m\}$, $N_{\delta_n}(Y_i) \geq 2^{\Omega(n)}$.*

Step 3 In the third step, we combine Proposition 1 with Lemma 3 to obtain a non-linear lower bound for the size of Boolean formulas computing the element distinctness function $\delta_n : \{0,1\}^X \to \{0,1\}$ defined in (1). Let F be a Boolean formula computing δ_n. Let l_i denote the number of inputs of F labeled with some variable in Y_i. By Proposition 1, we have that $N_{\delta_n}(Y_i) \leq 2^{O(l_i)}$. On the other hand, by Lemma 3, $N_{\delta_n}(Y_i) \geq 2^{\Omega(n)}$. Combining these two inequalities, we have that

$$2^{O(l_i)} \geq N_{\delta_n}(Y_i) \geq 2^{\Omega(n)}. \quad (2)$$

This implies that $l_i \geq \Omega(n)$. In other words, there are $\Omega(n)$ inputs of F labeled with variables from Y_i. Since there are $m = \Omega(\frac{n}{\log n})$ blocks Y_i, we have that the number of inputs of F labeled with variables in X is at least $\Omega(\frac{n^2}{\log n})$.

3.1 Generalizing Nečiporuk's Theorem

In this section we will generalize Nečiporuk's non-linear lower bound to the context of circuits of low treewidth (Theorem 1). We call attention to the fact that this lower bound concerns circuits in which each variable can label arbitrarily many input vertices. The following lemma, which generalizes Proposition 1, is the main technical result toward the proof of Theorem 1.

Lemma 4 *Let $f : \{0,1\}^X \to \{0,1\}$ be a function computable by a Boolean circuit $\mathcal{C}$ of treewidth t. Let $Y \subseteq X$ be a subset of variables such that at most l inputs of $\mathcal{C}$ are labeled with variables in Y. Then $N_f(Y)$ is at most $2^{l \cdot 2^{O(t)}}$.*

The next two subsections will be dedicated to the proof of Lemma 4. Before, we show how Lemma 4 can be used to prove Theorem 1.

Proof of Theorem 1 Let $|X| = n = 2m \log m$, and $Y_1, ..., Y_m$ be a partition of the variables in X, where for each i, $|Y_i| = 2 \log m$. Let l_i be the number of inputs of $\mathcal{C}$ labeled with a variable from Y_i. By Lemma 3, $N_{\delta_n}(Y_i) \geq 2^{\Omega(n)}$. On the other hand, by Lemma 4, $N_{\delta_n}(Y_i) \leq 2^{l_i \cdot 2^{O(t)}}$. Therefore, by combining these two inequalities, we have $2^{l_i \cdot 2^{O(t)}} \geq N_{\delta_n}(Y_i) \geq 2^{\Omega(n)}$. This implies that $l_i \geq \Omega(n/2^{O(t)})$. Since there are

$m = \Omega(\frac{n}{\log n})$ blocks of variables Y_i, we have that the number of inputs of $\mathcal{C}$ is at least $\frac{n^2}{2^{O(t)} \cdot \log n}$. □

3.2 Defining Relations via Constraint Satisfaction Problems

In this subsection we will introduce some terminology and basic results which will be used in the proof of Lemma 4. Let X be a set of variables. An *assignment* of X is a function $a : X \to \{0, 1\}$ that associates with each variable $x \in X$ a value $a(x) \in \{0, 1\}$. We let $\{0, 1\}^X$ denote the set of all assignments of X. A *relation* over X is any subset $R \subseteq \{0, 1\}^X$. We say that each variable $x \in X$ is *constrained* by R. In some places we write $var(R)$ to denote the set of variables constrained by R. If a is an assignment of X, and $Y \subseteq X$, then we let $a|_Y$ denote the *restriction* of a to Y. More precisely, for each $x \in Y$, $a|_Y(x) = a(x)$. We say that an assignment $a \in \{0, 1\}^X$ *satisfies a relation* R over $Y \subseteq X$ if $a|_Y \in R$. If $R \subseteq \{0, 1\}^X$ is a relation over X, and $Y \subseteq X$, then the restriction of R to Y is the relation $R|_Y = \{a|_Y \mid a \in R\}$. The following immediate observation states that the result of restricting a relation $R \subseteq \{0, 1\}^X$ to a subset X' and subsequently to a subset $Y \subseteq X'$ is equivalent to restricting R directly to Y.

Observation 1 *Let R be a relation over X and let $Y \subseteq X' \subseteq X$. Then $R|_Y = (R|_{X'})|_Y$.*

Below, we define the notion of *constraint satisfaction problem* over X.

Definition 3 A constraint satisfaction problem (CSP) over a set of variables X is a set of relations

$$K = \{R_1, R_2, ..., R_r\} \tag{3}$$

where for each $i \in \{1, ..., r\}$, R_i is a relation over some subset $X_i \subseteq X$ of variables.

A CSP K over a set of variables X can be used to define a relation $R(K)$ over X. Intuitively, the relation $R(K)$ consists of all assignments over X that satisfy each relation in K.

$$R(K) = \{a \in \{0, 1\}^X \mid a|_{X_i} \in R_i \text{ for } i \in \{1, ..., r\}.\} \tag{4}$$

Let K be a CSP over a set of variables X and let $S \subseteq K$. We denote by $c(S)$ the set of variables that are simultaneously constrained by some relation in S and some relation $K \backslash S$. We say that $c(S)$ is the *cutset* of S with respect to K. Given a CSP K over a set of variables X, and a subset $Y \subseteq X$, we will deal with the problem of obtaining a CSP K' with less relations than K, but with the property that $R(K)|_Y = R(K')|_Y$. The following simple lemma will be crucial for this goal.

Lemma 5 *Let $K = \{R_1, ..., R_r\}$ be a CSP over a set of variables X, let $Y \subseteq X$, and $S \subseteq K$ be such that $var(S) \cap Y \subseteq c(S)$. Consider the CSP*

$$K' = (K \backslash S) \cup \{R(S)|_{c(S)}\}.$$

Then $R(K)|_Y = R(K')|_Y$.

Proof First we note that K' is a CSP over the set of variables $X' = (X \backslash var(S)) \cup c(S)$. Second, we note that the relation induced by K' on X' is precisely the restriction of $R(K)$ to the variables in X'. In other words, $R(K)|_{X'} = R(K')$. Finally, since $Y \subseteq X$ and $var(S) \cap Y \subseteq c(S)$, we have that $Y \subseteq X'$. Therefore, by Observation 1, $R(K)|_Y = (R(K)|_{X'})|_Y = R(K')|_Y$. □

3.3 Circuits vs CSPs

In this section we prove Lemma 4. The idea behind the proof is the following. Let $\mathcal{C} = (V, E, \mathbf{g})$ be a circuit of carving width w computing a function $f : \{0,1\}^Y \to \{0,1\}$. As a first step, we associate with $\mathcal{C}$ a CSP $K(\mathcal{C})$ over a set of variables $Y \cup \{x_e \mid e \in E\}$. This CSP has the property that $R(K(\mathcal{C}))|_Y$ consists precisely of those assignments that cause $\mathcal{C}$ to evaluate to 1. In a second step, we use the fact that $\mathcal{C}$ has carving width w to obtain a new CSP K' such that each relation in K' constrains at most $2 \cdot w$ variables, and such that the number of relations in K' is proportional to the number of uninitialized inputs of $\mathcal{C}$. This new CSP K' has the property that $R(K')|_Y = R(K(\mathcal{C}))|_Y$. Finally, if we are given a function $f : \{0,1\}^X \to \{0,1\}$ and a subset $Y \subseteq X$ of variables with $|Y| = l$, then we will have that there are at most $2^{l \cdot 2^{O(w)}}$ distinct functions arising by restricting all variables in $X \backslash Y$ to values in $\{0,1\}$.

Definition 4 (CSP Derived from a Circuit) Let $\mathcal{C} = (V, E, \mathbf{g})$ be a circuit whose inputs are labeled with variables from Y. We let $K(\mathcal{C}) = \{R_v \mid v \in V\}$ be the CSP over the variables $Y \cup \{x_e \mid e \in E\}$ which is defined as follows.

1. If v is an input vertex labeled by $\mathbf{g}$ with a variable x, and v is the source of edges $e_1, ..., e_k$, then R_v is a relation over the variables $Y_v = \{x, x_{e_1}, ..., x_{e_k}\}$, and an assignment $a : Y_v \to \Sigma$ is in R_v if and only if
$$a(x) = a(x_{e_1}) = ... = a(x_{e_k}).$$
2. If v is an internal vertex labeled with a gate $\mathbf{g}(v)$, v is the target of edges $e_1, ...e_k$, and v is the source of edges $e'_1, ..., e'_{k'}$, then R_v is a relation over the variables $Y_v = \{x_{e_1}, ..., x_{e_k}, x_{e'_1}, ..., x_{e'_{k'}}\}$, and an assignment $a : Y_v \to \Sigma$ is in R_v if and only if
$$\mathbf{g}(v)(a(x_{e_1}), ..., a(x_{e_k})) = a(x_{e'_1}) = ... = a(x_{e'_{k'}}).$$
3. If v is the output vertex of $\mathcal{C}$, and v is the target of edges $e_1, ..., e_k$, then R_v is a relation over the variables $Y_v = \{x_{e_1}, ..., .x_{e_k}\}$, and $a : \{x_{e_1}, ..., x_{e_k}\} \to \Sigma$ is in R_v if and only if
$$\mathbf{g}(v)(x_{e_1}, ..., x_{e_k}) = 1.$$

Intuitively, the variables Y are input variables of the circuit $\mathcal{C}$, while the variables $\{x_e \mid e \in E\}$ are used to keep track of the evaluation of the circuit $\mathcal{C}$ when the variables in Y are initialized. The relation $R(K(\mathcal{C}))$ associated with the CSP $K(\mathcal{C})$

contains all assignments of $Y \cup \{x_e \mid e \in E\}$ which encode an initialization of the input variables together with an evaluation of the gates of the circuits which evaluate to 1. If we restrict the relation $R(K(\mathcal{C}))$ to the variables in Y, then we recover precisely the set of assignments that cause $\mathcal{C}$ to evaluate to 1.

Observation 2 *Let $\mathcal{C}$ be a circuit computing a function $f : \{0,1\}^Y \to \{0,1\}$. Let $K(\mathcal{C})$ be the CSP associated with $\mathcal{C}$. Then $R(K(\mathcal{C}))|_Y = \{a \in \{0,1\}^Y \mid f(a) = 1\}$.*

We note that the number of relations in $R(K(\mathcal{C}))$ is precisely the number of gates of $\mathcal{C}$, and therefore there is no a priori correspondence between the number of relations in $R(K(\mathcal{C}))$ and the number of inputs of $\mathcal{C}$ labeled by variables in Y. The following theorem says that if $\mathcal{C}$ is a circuit of carving width w then one can construct a CSP K whose size is proportional to the number of inputs of $\mathcal{C}$ labeled with variables in Y, in such a way that the number of variables constrained by each relation in K is proportional to w, and such that $R(K)|_Y = R(K(\mathcal{C}))|_Y$.

Theorem 2 (CSP Reduction) *Let $\mathcal{C}$ be a circuit of carving width w computing a function $f : \{0,1\}^Y \to \{0,1\}$. Let $l \geq |Y|$ be the number of inputs of $\mathcal{C}$ labeled with variables in Y. Then there exits a CSP $K = \{R_1, ..., R_k\}$ with $k \leq 3 \cdot l$ such that for each $i \in \{1, ..., k\}$, R_i constrains at most $2 \cdot w$ variables and such that $R(K)|_Y = \{a \in \{0,1\}^Y \mid f(a) = 1\}$.*

It is worth noting that the CSP K in Theorem 2 is obtained from the CSP $K(\mathcal{C})$ by applying several non-trivial simplification steps. Before proving Theorem 2 we show how this theorem can be used to prove Lemma 4.

Proof of Lemma 4 Let $f : \{0,1\}^X \to \{0,1\}$ be a Boolean function which is computable by a circuit $\mathcal{C}$ of treewidth t. By Lemma 2, there exists a circuit $\mathcal{C}'$ of size $|\mathcal{C}'| \leq O(|\mathcal{C}|)$, maximum degree 3 and carving width $w = O(t)$ such that $\mathcal{C}'$ computes f. Additionally, for each variable x, the number of input vertices labeled with x in $\mathcal{C}$ is equal to the number of input vertices labeled with x in $\mathcal{C}'$.

Now let $Y \subseteq X$. Then each initialization $b \in \{0,1\}^{X \setminus Y}$ of the variables in $X \setminus Y$, gives rise to a circuit $\mathcal{C}_b$ in which all l uninitialized inputs are labeled with variables in Y. Let $f_b : \{0,1\}^Y \to \{0,1\}$ be the function computed by the circuit $\mathcal{C}_b$. By Theorem 2, there is a CSP $K_b = \{R_1^b, ..., R_{k_b}^b\}$ with $k_b \leq 3 \cdot l$ such that for each $i \in \{1, ..., k_b\}$, R_i^b constrains at most $2 \cdot w = O(t)$ variables and such that $R(K_b)|_Y = \{a \in \{0,1\}^Y \mid f_b(a) = 1\}$.

We note that by the way in which the CSP K_b is constructed in the proof of Theorem 2, the set of variables constrained by each relation R_i^b does not depend on the initialization b. In other words, for each two distinct initializations $b, b' \in \{0,1\}^{X \setminus Y}$, we have that $k_b = k_{b'}$ and that R_i^b constrains the same variables as $R_i^{b'}$ for each $i \in \{1, ..., k_b\}$.

Therefore, since there are at most $2^{2^{O(t)}}$ possible relations for each $i \in \{1, ..., k_b\}$, we can conclude that there are at most $2^{l \cdot 2^{O(t)}}$ distinct functions which can be obtained from f by restricting the variables outside Y. In other words, $N_f(Y) \leq 2^{l \cdot 2^{O(t)}}$. □

In the remainder of this subsection, we prove Theorem 2. Let $\mathcal{C} = (V, E, \mathbf{g})$ be a circuit of carving-width w computing a function $f : \{0,1\}^Y \to \{0,1\}$. Let $K = K(\mathcal{C}) = \{R_v \mid v \in V\}$ be the CSP associated with $\mathcal{C}$. We note that $Y \cup \{x_e \mid e \in E\}$ is the set of variables constrained by relations in K. We say that a relation R_v is a Y-relation if R_v constrains some variable in Y. Let (T, γ) be a carving decomposition of (V, E). For a node u of T we let $leaves(T[u], Y)$ denote the set of leaves u' of $T[u]$ such that the relation $R_{\gamma(u')}$ is a Y-relation. We say that a node $u \in nodes(T)$ is a Y-node if u is either a leaf such that $R_{\gamma(u)}$ is a Y-relation, or if u is an internal node $u \in nodes(T)$ such that $leaves(T[u.l], Y) \neq \emptyset$ and $leaves(T[u.r], Y) \neq \emptyset$. If u is a Y-node, then we say that a node $u' \neq u$ is the Y-parent of u if u' is the ancestor of u at minimal distance from u with the property that u' is itself a Y-node. We let $nodes(T, Y)$ denote the set of all Y-nodes of T.

Lemma 6 $|nodes(T, Y)| = 2 \cdot |leaves(T, Y)| - 1$.

Proof First, we show that if u is an internal Y-node of T then u has precisely two Y-children. Suppose that u has at most one Y-child. Then by definition u is not a Y-node, since in this case either $leaves(T[u.l], Y) = \emptyset$ or $leaves(T[u.r], Y) = \emptyset$. Now suppose that u has at least 3 Y-children. Since T is a binary tree, two Y-children of u are either descendants of $u.l$ or descendants of $u.r$. Lets assume that z and z' are two distinct Y-children of u which are also descendants of $u.l$. We observe that neither z is a descendant of z' nor z' is a descendant of z, since otherwise, only one of these two vertices could have been a Y-child of u. Now let u' be the closest ancestor of z which is also an ancestor of z'. Then u' is by definition a Y-node. Since u' is a strict descendant of u, this contradicts the assumption that u is the Y-parent of z and z'.

Now let $T[Y]$ be the tree whose nodes are Y-nodes of T and such that (u, v) is an arc of $T[Y]$ if and only if u is the Y-parent of v. Then by the discussion above we have that $T[Y]$ is a binary tree with $|leaves(T, Y)|$ leaves. Since any binary tree with l leaves has $l - 1$ internal nodes, the total number of Y-nodes in T is $2 \cdot |leaves(T, Y)| - 1$. □

Now let $T' = T \backslash nodes(T, Y)$ be the forest which is obtained by deleting from T all of its Y-nodes. Then the number of connected components in the forest T' is at most $|nodes(T, Y)| = 2|leaves(T, Y)| - 1$. We let $T_1, ..., T_k$, for $k \leq |nodes(T, Y)|$ be the connected components of T'. For each $i \in \{1, ..., k\}$, let

$$S_i = \{R_v \mid \exists u \in leaves(T_i),\ \gamma(u) = v\}$$

be the sub-CSP of $K(\mathcal{C})$ formed by the relations associated to vertices of $\mathcal{C}$ that label the leaves of the connected component T_i. Let $c(S_i) = var(K(\mathcal{C}) \backslash S_i) \cap var(S_i)$ be the cut-set of S_i with respect to $K(\mathcal{C})$. In other words, $c(S_i)$ is the set of variables that are constrained by some relation in S_i, and another relation in $K \backslash S_i$. Note that $c(S_i) \cap Y = \emptyset$, since the connected component T_i has no Y-node. Additionally, the fact that (T, γ) is a carving decomposition of $\mathcal{C}$ of width w implies the following claim.

Claim The number of variables in $c(S_i)$ is at most $2 \cdot w$.

Let $R_i = R(S_i)|_{c(S_i)}$. Then we define our CSP as follows.

$$K = \left(K(\mathcal{C}) \setminus \bigcup_{i=1}^{k} S_i \right) \cup \bigcup_{i=1}^{k} \{R_i\} \tag{5}$$

Note that each subset of relations $S_i \subseteq K(\mathcal{C})$ corresponding to the connected component T_i is replaced by a unique relation R_i. By Lemma 6, there are at most $2 \cdot |leaves(T, Y)| - 1$ connected components in T'. Therefore, the number of relations in K is upper-bounded by $|leaves(T, Y)| + 2|leaves(T, Y)| - 1 < 3|leaves(T, Y)|$. We claim that $R(K)|_Y = R(K(\mathcal{C}))|_Y$. To prove this claim, let $K_0, K_1, ..., K_k$ be a sequence of CSPs where $K_0 = K(\mathcal{C})$, and for each $i \in \{1, ..., k\}$, $K_i = (K_{i-1} \setminus S_i) \cup \{R_i\}$. Then clearly we have that $K = K_k$. We claim that for each $j \in \{0, ..., k\}$, $K_j|_Y = K(\mathcal{C})|_Y$. In the base case $k = 0$, and the claim follows trivially. Now assume that $K_j|_Y = K(\mathcal{C})|_Y$. By Lemma 5, we have that $K_j|_Y = K_{j+1}|_Y$.

4 Symmetric Non-deterministic State Complexity

Let Σ be a finite set of symbols. In this section we define the notion of symmetric non-deterministic state complexity of functions of the form $f : \Sigma^n \to \{0, 1\}$ and of finite languages included in Σ^n. We note that this notion is polynomially related with the size of the smallest non-deterministic oblivious, read-once branching program [35] computing f. A non-deterministic finite automaton (NFA) over Σ is a 5-tuple $\mathcal{A} = (Q, \Sigma, \mathfrak{R}, Q_0, F)$ where Q is a set of states, $Q_0 \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of final states and $\mathfrak{R} \subseteq Q \times \Sigma \times Q$ is a transition relation. We write $q \xrightarrow{a} q'$ to denote that the triple (q, a, q') belongs to $\mathfrak{R}$. We say that a string $w = w_1 w_2 ... w_n \in \Sigma^n$ is accepted by $\mathcal{A}$ if there is a sequence $q_0 \xrightarrow{w_1} q_1 \xrightarrow{w_2} ... \xrightarrow{w_n} q_n$ such that $q_0 \in Q_0$ and $q_n \in Q_F$. We denote by $\mathcal{L}(\mathcal{A})$ the set of all strings accepted by $\mathcal{A}$.

Let $\mathcal{L} \subseteq \Sigma^n$ be a set of length-n strings over Σ. The *non-deterministic* state complexity (NSC) of $\mathcal{L}$, denoted $nsc(\mathcal{L})$, is defined as the minimum number of states of a NFA accepting $\mathcal{L}$. Let $f : \Sigma^n \to \{0, 1\}$ be a function. We denote by $\mathcal{L}(f)$ the set of all strings $w \in \Sigma^n$ for which $f(w) = 1$. We define the non-deterministic state complexity of f as $nsc(f) := nsc(\mathcal{L}(f))$.

For each positive integer n, we let $Perm(n)$ be the set of all permutations of the set $[n] = \{1, ..., n\}$. If $\pi : [n] \to [n]$ is a permutation in $Perm(n)$ and $w \in \Sigma^n$, then we let $\pi(w)$ be the string in Σ^n that is defined by setting $\pi(w)_{\pi(j)} = w_j$ for each $j \in [n]$. Intuitively, the j-th position of w is mapped to the position $\pi(j)$ of $\pi(w)$. If $\mathcal{L} \subseteq \Sigma^n$ is a set of length-n strings over Σ, then we denote by $\pi(\mathcal{L})$ the language obtained from $\mathcal{L}$ by permuting the coordinates of each string in $\mathcal{L}$ according to π. More precisely,

$$\pi(\mathcal{L}) = \{\pi(w) \mid w \in \mathcal{L}\}. \tag{6}$$

The symmetric non-deterministic state complexity (symmetric-NSC) of a language $\mathcal{L} \subseteq \Sigma^n$ is defined as the minimum non-deterministic state complexity of a permuted version of $\mathcal{L}$.

Definition 5 (Symmetric Nondeterministic State Complexity) Let $\mathcal{L} \subseteq \Sigma^n$. The symmetric non-deterministic state complexity of $\mathcal{L}$ is defined as

$$snsc(\mathcal{L}) = \min_{\pi \in Perm(n)} nsc(\pi(\mathcal{L})). \tag{7}$$

The symmetric-NSC of a function $f : \Sigma^n \to \{0,1\}$ is defined as $snsc(f) = snsc(\mathcal{L}(f))$.

We note that the symmetric-NSC of a function $f : \{0,1\}^n \to \{0,1\}$ is lower-bounded by the size of the smallest non-deterministic oblivious read-once $|\Sigma|$-way branching program computing f. Therefore, functions requiring exponential size branching programs of this particular form have exponential symmetric non-deterministic state complexity. Two examples of such functions are the *triangle-freeness* function $\Delta_n : \{0,1\}^n \to \{0,1\}$, and the *triangle-parity* function $\bigoplus \text{Clique}_{3,n} : \{0,1\}^n \to \{0,1\}$. Both functions take as input an array $x = (x_{ij})_{1 \le i < j \le m}$ consisting of $n = \binom{m}{2}$ Boolean variables representing an undirected graph $G(x)$ on m vertices $\{1, ..., m\}$. The graph $G(x)$ has an edge connecting vertices i and j, with $i < j$, if and only if $x_{ij} = 1$. The triangle-freeness function Δ_n returns 1 on an input x if and only if the graph $G(x)$ does not contain a triangle. The triangle-parity function $\bigoplus \text{Clique}_{3,n}$ returns 1 if and only if the parity of the number of the triangles in $G(x)$ is odd. Building on techniques from communication complexity theory [28], it can be shown that read-once non-deterministic branching programs computing the functions Δ_n and $\bigoplus \text{Clique}_{3,n}$ require size $2^{\Omega(n)}$ [11]. Therefore, the same lower bound holds for the symmetric-NSC of these functions.

Theorem 3 ([11]) *$snsc(\Delta_n) \ge 2^{\Omega(n)}$ and $snsc(\bigoplus \text{Clique}_{3,n}) \ge 2^{\Omega(n)}$.*

4.1 On a Variant of the Element Distinctness Function

We say that a binary string w is *even* if w has an even number of ones. Analogously, we say that w is *odd* if w has an odd number of ones. For each $r \in \mathbb{N}$, we let $even(r)$ denote the set of all strings of even parity in the set $\{0,1\}^r$. Let $\Sigma(m) = \{1, ..., m\}$, and $P(m) \subseteq \Sigma(m)^m$ be the set of all length-m strings over $\Sigma(m) = \{1, ..., m\}$ whose entries are pairwise distinct. Let $n = (\lceil \log m \rceil + 1) \cdot m$ and let $b : \{1, ..., m\} \to even(\lceil \log m \rceil + 1)$ be an injection that maps each number $j \in \{1, ..., m\}$ to an even binary string $b(j)$ of length $\lceil \log m \rceil + 1$. We let

$$B(n) = \{b(w_1)b(w_2)...b(w_m) \in \{0,1\}^n \mid w_1 w_2 ... w_m \in P(m)\}$$

be the binary language that is obtained from $P(m)$ by mapping each string in $P(m)$ to its binary representation. We define the *even element distinctness function* $\tau_n : \{0,1\}^n \to \{0,1\}$ as the function that returns 1 on an input $w \in \{0,1\}^n$ if and only if $w \in B(n)$. Note that by definition, the symmetric-NSC of τ_n is the symmetric-NSC of $B(n)$.

Theorem 4 *The function $\tau_n : \{0,1\}^n \to \{0,1\}$ has symmetric-NSC $2^{\Omega(n/\log n)}$.*

The proof of Theorem 4 will use the following result.

Theorem 5 (Glaister-Shallit [15]) *Let $\mathcal{L} \subseteq \Sigma^n$ be a set of length-n strings over Σ, and suppose that there exists a set $F = \{(x_i, w_i) \mid 1 \leq i \leq k\}$ of pairs of strings such that*

1. $x_i \cdot w_i \in \mathcal{L}$ *for* $1 \leq i \leq k$
2. $x_i \cdot w_j \notin \mathcal{L}$ *for* $1 \leq i, j \leq k$ *and* $i \neq j$

Then any non-deterministic finite automaton accepting $\mathcal{L}$ has at least k states.

The set F in Theorem 5 is called a fooling set for $\mathcal{L}$. We will prove Theorem 4 by constructing, for each permutation $\pi : [n] \to [n]$, a fooling set F_π of size $2^{\Omega(n/\log n)}$ for the language $\pi(B(n))$. To construct F_π it will be convenient to view strings as Boolean functions over sets of positions. In other words, if S is a set of positive integers, then a string over S is simply a Boolean function $w : S \to \{0, 1\}$. We note that we allow S to be any set of positive integers and not necessarily an interval of the form $[n] = \{1, ..., n\}$. The parity of w is defined as the parity of the number of positions in which w evaluates to 1: $par(w) = |\{i \in S \mid w(i) = 1\}| \mod 2$. The restriction of w to a subset $T \subseteq S$ is the string $w|_T : T \to \{0, 1\}$ which is defined by setting $w|_T(i) = w(i)$ for every $i \in T$.

If $S \subseteq [n]$, $w : S \to \{0, 1\}$ is a string, and $\pi : [n] \to [n]$ is a permutation, then we let $\pi(w)$ be the string $w' : \pi(S) \to \{0, 1\}$ that is defined by setting $w'(\pi(i)) = w(i)$ for each $i \in S$. We let $L = \{1, ..., \lfloor n/2 \rfloor\}$ and $R = \{\lfloor n/2 \rfloor + 1, ..., n\}$ be respectively the first and the second halves of the set $[n] = \{1, ..., n\}$. We say that a permutation $\pi : [n] \to [n]$ splits a subset $S \subseteq [n]$ if $\pi(S) \cap L \neq \emptyset$ and $\pi(S) \cap R \neq \emptyset$. In other words, π splits S if some elements of S are mapped by π to the first half of $[n]$ and some elements of S are mapped by π to the second half of $[n]$.

If S and S' are subsets of $[n]$ such that $S \cap S' = \emptyset$, and $w : S \to \{0, 1\}$ and $w' : S' \to \{0, 1\}$ are strings with domain S and S' respectively, then the concatenation of w with w' is simply the function $w \cdot w' : S \cup S' \to \{0, 1\}$ which is equal to w when restricted to S and equal to w' when restricted to S'.

Lemma 7 *Let S be a subset of $[n] = \{1, ..., n\}$. Let $\mathcal{L} \subseteq \{0, 1\}^n$ be a set of strings with domain $[n]$ such that $w|_S$ is even for every $w \in \mathcal{L}$. Let $\pi : [n] \to [n]$ be a permutation, and w_1 and w_2 be binary strings with domain $[n]$ such that*

1. *S is split by π,*
2. *$\pi(w_1)|_{\pi(S)\cap L}$ is even and $\pi(w_1)|_{\pi(S)\cap R}$ is even.*
3. *$\pi(w_2)|_{\pi(S)\cap L}$ is odd and $\pi(w_2)|_{\pi(S)\cap R}$ is odd.*

Then $[\pi(w_1)|_L] \cdot [\pi(w_2)|_R] \notin \pi(\mathcal{L})$ and $[\pi(w_2)|_L] \cdot [\pi(w_1)|_R] \notin \pi(\mathcal{L})$.

Proof Let $w = [\pi(w_1)|_L] \cdot [\pi(w_2)|_R]$. We show that $w \notin \pi(\mathcal{L})$. The proof is by contradiction. Assume that $w \in \pi(\mathcal{L})$. Since $\left[\pi(w_1)|_{\pi(S)\cap L}\right]$ is even and $\left[\pi(w_2)|_{\pi(S)\cap R}\right]$ is odd, we have that $w|_{\pi(S)}$ is odd. But by definition, $w \in \pi(\mathcal{L})$ if and only if there exists $w' \in \mathcal{L}$ such that $w = \pi(w')$. Since by assumption $w'|_S$ is even for every $w' \in \mathcal{L}$, we have that $\pi(w')|_{\pi(S)} = w|_{\pi(S)}$ is also even. Thus

we have reached a contradiction and we conclude that $w \notin \pi(\mathcal{L})$. The proof that $\left[\pi(w_2)|_{\pi(S)\cap L}\right] \cdot \left[\pi(w_1)|_{\pi(S)\cap R}\right] \notin \mathcal{L}$ is analogous. □

Let $S = \{j_1, j_2, ..., j_k\} \subseteq [n]$ where $j_1 < j_2 < ... < j_k$. For $i \in \{1, ..., k\}$, we let $S[i] = j_i$ denote the i-th element of S. Let S and S' be subsets of $[n]$ of same size. We say that strings $w : S \to \{0, 1\}$ and $w' : S' \to \{0, 1\}$ are equivalent, which we denote by $w \equiv w'$, if for each $i \in \{1, ..., |S|\}$, $w(S[i]) = w(S'[i])$. Note that if $S = S'$ then $w \equiv w'$ if and only if $w = w'$.

Let $n = (1 + \lceil \log m \rceil) \cdot m$, and let $I_1, ..., I_m$ be the sequence of subsets of $[n]$ such that for each $i \in [m]$,

$$I_i = \{(i - 1) \cdot (1 + \lceil \log m \rceil) + 1, ..., i \cdot (1 + \lceil \log m \rceil)\}.$$

In other words, $I_1, ..., I_m$ is a partition of the set $[n]$ into m consecutive intervals of equal size. We say that $I_1, ..., I_m$ is the *uniform interval partition* of $[n]$.

Lemma 8 *Let $\pi : [n] \to [n]$ be a permutation where $n = (\lceil \log m \rceil + 1) \cdot m$, and let $I_1, ..., I_m$ be the uniform interval partition of $[n]$. Let w_1 and w_2 be strings in $B(n)$ such that for some $i, j \in [m]$, with $i \neq j$, the following conditions are satisfied.*

1. $\pi(I_i) \subseteq L$ *and* $\pi(I_j) \subseteq R$.
2. $w_1|_{I_i} \equiv w_2|_{I_j}$.
3. $w_1|_{I_j} \equiv w_2|_{I_i}$.

Then $[\ \pi(w_1)|_L\ ] \cdot [\ \pi(w_2)|_R\ ] \notin \pi(B(n))$ *and* $[\ \pi(w_2)|_L\ ] \cdot [\ \pi(w_1)|_R\ ] \notin \pi(B(n))$.

Proof Let $w = [\ \pi(w_1)|_L\ ] \cdot [\ \pi(w_2)|_R\ ]$. We show that $w \notin \pi(B(n))$. The proof is by contradiction. Assume that $w \in \pi(B(n))$. Then there is a string $w' \in B(n)$ such that $w = \pi(w')$. Since by assumption, $\pi(I_i) \subseteq L$ and $\pi(I_j) \subseteq R$,

$$w|_{\pi(I_i)} = \pi(w_1)|_{\pi(I_i)} = \pi(w_1|_{I_i}) \qquad \text{and} \qquad w|_{\pi(I_j)} = \pi(w_2)|_{\pi(I_j)} = \pi(w_2|_{I_j}).$$

Now let π^{-1} be the inverse of π. Then, $w' = \pi^{-1}(w)$. Additionally,

$$w'|_{I_i} = \pi^{-1} \circ \pi(w_1|_{I_i}) = w_1|_{I_i} \qquad \text{and} \qquad w'|_{I_j} = \pi^{-1} \circ \pi(w_2|_{I_j}) = w_2|_{I_j}.$$

But by assumption, $w_1|_{I_i} \equiv w_2|_{I_j}$, and therefore, $w'|_{I_i} \equiv w'|_{I_j}$. This contradicts the assumption that $w' \in B(n)$, since for each string $w' \in B(n)$ and each $i, j \in [m]$ with $i \neq j$, we must have $w'|_{I_i} \not\equiv w'|_{I_j}$ (by definition of $B(n)$). The proof that $[\ \pi(w_2)|_L\ ] \cdot [\ \pi(w_1)|_R\ ] \notin B(n)$ is analogous. □

Let $n = (\lceil \log m \rceil + 1) \cdot m$, $\pi : [n] \to [n]$ be a permutation of $[n]$, and $I_1, ..., I_m$ be the uniform interval partition of $[n]$. Let $I_{i_1}, ..., I_{i_k}$, $I_{\ell_1}, ..., I_{\ell_l}$ and $I_{r_1}, ..., I_{r_l}$ be pairwise distinct intervals of this partition satisfying the following conditions.

1. I_{i_j} is split by π for every $j \in \{1, ..., k\}$.
2. $\pi(I_{\ell_j}) \subseteq L$ and $\pi(I_{r_j}) \subseteq R$ for each $j \in \{1, ..., l\}$.

Let $a_1, ..., a_k, b_1, ..., b_k, c_1, ..., c_l, d_1, ..., d_l$ be a sequence of pairwise inequivalent even strings satisfying the following conditions.

1. $a_j, b_j : I_{i_j} \to \{0, 1\}$ are strings with domain I_{i_j}.

2. $c_j : I_{\ell_j} \to \{0,1\}$ are strings with domain I_{ℓ_j}.
3. $d_j : I_{r_j} \to \{0,1\}$ are strings with domain I_{r_j}.
4. $\pi(a_j)|_{\pi(I_{i_j})\cap L}$ is even and $\pi(a_j)|_{\pi(I_{i_j})\cap R}$ is even.
5. $\pi(b_j)|_{\pi(I_{i_j})\cap L}$ is odd and $\pi(b_j)|_{\pi(I_{i_j})\cap R}$ is odd.

Next, we define a set of 2^{k+l} strings which will give rise to a fooling set for $\pi(B(n))$ of size 2^{k+l}. For each $(k+l)$ tuple of bits $x \in \{0,1\}^k$, $y \in \{0,1\}^l$, let $w[xy]$ be any string in $B(n)$ satisfying the following properties.

1. For each $j \in \{1,...,k\}$, if $x_j = 0$ then $w[xy]|_{I_{i_j}} \equiv a_j$.
2. For each $j \in \{1,...,k\}$, if $x_j = 1$ then $w[xy]|_{I_{i_j}} \equiv b_j$.
3. For each $j \in \{1,...,l\}$, if $y_j = 0$ then $w[xy]|_{I_{\ell_j}} \equiv c_j$ and $w[xy]|_{I_{r_j}} \equiv d_j$.
4. For each $j \in \{1,...,l\}$, if $y_j = 1$ then $w[xy]|_{I_{\ell_j}} \equiv d_j$ and $w[xy]|_{I_{r_j}} \equiv c_j$.

In other words, for each $j \in \{1,...,k\}$, the substring of $w[xy]$ corresponding to the positions in I_{i_j} is either equivalent to a_j (if $x_j = 0$), or equivalent to b_j (if $x_j = 1$). Analogously, for each $j \in \{1,...,l\}$, either the substring corresponding to the positions $I_{\ell j}$ is equivalent to c_j and the substring corresponding to I_{r_j} is equivalent to d_j (if $y_j = 0$), or vice-versa (if $y_j = 1$). Now we let $F_\pi^{k,l}$ be the set of pairs of strings obtained by splitting the permuted version $\pi(w[xy])$ of each string $w[xy]$ into two parts of equal size.

$$F_\pi^{k,l} = \{\ (\ \pi(w[xy])|_L,\ \pi(w[xy])|_R\ )\ \mid\ x \in \{0,1\}^k, y \in \{0,1\}^l\} \tag{8}$$

Theorem 6 *The set $F_\pi^{k,l}$ defined in (8) is a fooling set for $\pi(B(n))$ of size 2^{k+l}.*

Proof It should be clear that, by construction, the set $F_\pi^{k,l}$ has 2^{k+l} pairs. Therefore, we just need to show that if xy and $x'y'$ are $(k+l)$-bit strings differing in at least one entry, then neither $(\pi(w[xy])|_L) \cdot (\pi(w[x'y'])|_R)$ belongs to $\pi(B(n))$, nor $(\pi(w[x'y'])|_L) \cdot (\pi(w[xy])|_R)$ belongs to $\pi(B(n))$.

Note that since $xy \neq x'y'$, we have that $x_j \neq x'_j$ for some $j \in \{1,...,k\}$, or $y_j \neq y'_j$ for some $j \in \{1,...,l\}$ (or both).

First, suppose that $x_j \neq x'_j$ for some $j \in \{1,...,k\}$. We may assume without loss of generality that $x_j = 0$ and $x'_j = 1$, since otherwise we may just swap the roles of the strings xy and $x'y'$. Since $w[xy]|_{I_{i_j}} \equiv a_j$, we have that $\pi(w[xy])|_{\pi(I_{i_j})\cap L}$ is even and $\pi(w[xy])|_{\pi(I_{i_j})\cap R}$ is even. Analogously, since $w[x'y']|_{I_{i_j}} \equiv b_j$, we have that $\pi(w[x'y'])|_{\pi(I_{i_j})\cap L}$ is odd and $\pi(w[x'y'])|_{\pi(I_{i_j})\cap R}$ is odd. Therefore, Lemma 7 implies that neither $(\pi(w[xy])|_L)\cdot(\pi(w[x'y'])|_R)$ is in $\pi(B(n))$, nor $(\pi(w[x'y'])|_L)\cdot(\pi(w[xy])|_R)$ is in $\pi(B(n))$.

Now suppose that $y_j \neq y'_j$ for some $j \in \{1,...,l\}$. We may assume without loss of generality that $y_j = 0$ and $y'_j = 1$, since otherwise, we may simply swap the roles of xy and $x'y'$. Under this assumption, $w[xy]|_{I_{\ell_j}} \equiv c_j$, $w[xy]|_{I_{r_j}} \equiv d_j$, $w[x'y']|_{I_{\ell_j}} \equiv d_j$ and $w[x'y']|_{I_{r_j}} \equiv c_j$. Therefore, since $\pi(I_{\ell_j}) \subseteq L$ and $\pi(I_{r_j}) \subseteq R$, Lemma 8 implies that neither $(\pi(w[xy])|_L) \cdot (\pi(w[x'y'])|_R)$ is in $\pi(B(n))$, nor $(w[x'y']|_L) \cdot (w[xy]|_R)$ is in $\pi(B(n))$. □

Next, we will show that one can always choose appropriate values for k and l in such a way that the fooling set $F_\pi^{k,l}$ has size at least $2^{m/4} = 2^{\Omega(n/\log n)}$. We start by stating the following proposition, which is a straightforward consequence of Hall's marriage theorem [10].

Proposition 2 *Let $G = (A \dot\cup B, E)$ be a bipartite graph with vertex partitions A and B be such that $|A| \leq |B|$. If each vertex in A has degree at least $|A|$ then G has a matching of size $|A|$.*

Proof The proof is by induction on the size of A. In the base case, in which $|A| = 1$, the proposition holds trivially. Now assume the proposition holds for every graph such that $|A| \leq n$. Let $G' = (A' \dot\cup B', E')$ be a graph such that $|A'| = n + 1$. Select an arbitrary edge v, u where $v \in A'$ and $u \in B'$. Let $G = (A \dot\cup B, E)$ be the graph obtained by deleting v and u together with all edges incident with these vertices. Then we have that each vertex in A has at least n neighbors. By the induction hypothesis, G has a matching $M = \{\{v_1, u_1\}, ..., \{v_n, u_n\}\}$. Then we have that $M \cup \{\{v, u\}\}$ is a matching for G'. □

Proposition 2 will be used to prove the following crucial lemma.

Lemma 9 *Let $n = m \cdot (\lceil \log m \rceil + 1)$, $\pi : [n] \to [n]$ be a permutation, and $I_1, ..., I_m$ be the uniform interval partition of $[n]$. Let $I_{i_1}, ..., I_{i_k}$ be the intervals that are split by π. If $k \leq m/4$ then there exists a sequence $a_1, ..., a_k, b_1, ..., b_k$ of pairwise inequivalent even strings where for each $j \in \{1, ..., k\}$, $a_j, b_j : I_{i_j} \to \{0, 1\}$ are strings with domain I_{i_j}, and the following conditions are satisfied.*

1. *$\pi(a_j)|_{\pi(I_{i_j})\cap L}$ is even and $\pi(a_j)|_{\pi(I_{i_j})\cap R}$ is even.*
2. *$\pi(b_j)|_{\pi(I_{i_j})\cap L}$ is odd and $\pi(b_j)|_{\pi(I_{i_j})\cap R}$ is odd.*

Proof Let $M = \{1, ..., \lceil \log m \rceil + 1\}$. If $s : M \to \{0, 1\}$ is a string with domain M, then for each $j \in \{1, ..., k\}$, we let $p_j^s : I_{i_j} \to \{0, 1\}$ be the string with domain I_{i_j} such that $p_j^s \equiv s$.

Let $\mathcal{G} = (A \dot\cup B, E)$ be a bipartite graph whose vertex set is partitioned into sets

$$A = \{\mathbf{a}_1, ..., \mathbf{a}_k, \mathbf{b}_1, ..., \mathbf{b}_k\} \qquad \text{and} \qquad B = \{v_s \mid s : M \to \{0, 1\} \text{ is even}\}.$$

For each $j \in \{1, ..., k\}$, we add the edge $\{\mathbf{a}_j, v_s\}$ to E if and only if $\pi(p_j^s)|_{\pi(I_{i_j})\cap L}$ is even and $\pi(p_j^s)|_{\pi(I_{i_j})\cap R}$ is even. Analogously, we add the edge $\{\mathbf{b}_j, v_s\}$ to E if and only if $\pi(p_j^s)|_{\pi(I_{i_j})\cap L}$ is odd and $\pi(p_j^s)|_{\pi(I_{i_j})\cap R}$ is odd. We note that since there are $2^{|\pi(I_{i_j})\cap L|-1}$ even strings with domain $\pi(I_{i_j}) \cap L$, and $2^{|\pi(I_{i_j})\cap R|-1}$ even strings with domain $\pi(I_{i_j}) \cap R$, we have that there are $2^{|\pi(I_{i_j})|-2} \geq m/2$ strings which are even in both $\pi(I_{i_j}) \cap L$ and in $\pi(I_{i_j}) \cap R$. Therefore, the vertex $\mathbf{a}_j$ has at least $m/2$ neighbors. Analogously, the vertex $\mathbf{b}_j$ has at least $m/2$ neighbors. Since $k \leq m/4$, we have that $|A| \leq m/2$. Therefore, by Proposition 2, G has a matching $\mathcal{M}$ with k edges. Now, for each $j \in \{1, ..., k\}$, we set $a_j = p_j^s$ if and only if the

edge $\{\mathbf{a}_j, v_s\}$ belongs to $\mathcal{M}$. Analogously, we set $b_j = p_j^s$ if and only if the edge $\{\mathbf{b}_j, v_s\} \in \mathcal{M}$. Since $\mathcal{M}$ is a matching, all strings $a_1, ..., a_k, b_1, ..., b_k$ are pairwise inequivalent. □

Now, assume that the number k of intervals split by π is less than $m/4$. The following proposition states that there are plenty of pairs of the form (I_{ℓ_j}, I_{r_j}) such that $\pi(I_{\ell_j}) \subseteq L$ and $\pi(I_{r_j}) \subseteq R$.

Proposition 3 *Let $n = m \cdot (\lceil \log m \rceil + 1)$, $\pi : [n] \to [n]$ be a permutation, and $I_1, ..., I_m$ be the uniform interval partition $[n]$. Let k be the number of intervals in this partition that are split by π. If $k < m/2$, then there exist at least $l = m/2 - 1 - k$ pairs of integers $(\ell_1, r_1), (\ell_2, r_2), ..., (\ell_l, r_l)$ such that $\pi(I_{\ell_j}) \subseteq L$ and $\pi(I_{r_j}) \subseteq R$ for every $j \in \{1, ..., l\}$.*

Proof Recall that $L = \{1, ..., \lfloor n/2 \rfloor\}$ and $R = \{\lfloor n/2 \rfloor + 1, ..., n\}$. Let $I_{i_1}, ..., I_{i_k}$ be the intervals that are split by π. Let $L' = L \setminus \bigcup_{j=1}^{k} \pi(I_{i_j})$ and $R' = R \setminus \bigcup_{j=1}^{k} \pi(I_{i_j})$. Since $k \leq m/2 - 1$, and since $|I_{i_j}| = (\lceil \log m \rceil + 1)$ for each $j \in \{1, ..., k\}$, we have that $|y| \geq (\frac{m}{2} - 1 - k)(\lceil \log m \rceil + 1)$ and that $|R'| \geq (\frac{m}{2} - 1 - k)(\lceil \log m \rceil + 1)$. Now we have that for each position $q \in L'$, there must exist an interval I_s which is not split by π such that $q \in \pi(I_s) \subseteq L$. This implies that there must exist at least $l = m/2 - 1 - k$ intervals $I_{\ell_1}, ..., I_{\ell_l}$ such that $\pi(I_{\ell_j}) \subseteq L$ for each $j \in \{1, ..., l\}$. By a similar reasoning, there exists l intervals $I_{r_1}, ..., I_{r_l}$ such that $\pi(I_{r_j}) \subseteq R$ for every $j \in \{1, ..., l\}$. □

By combining Lemma 9 with Proposition 3, we have the following theorem.

Theorem 7 *Let $n = m \cdot (\lceil \log m \rceil + 1)$, $\pi : [n] \to [n]$ be a permutation, and $I_1, ..., I_m$ be the uniform interval partition $[n]$. Let k be the number of intervals split by π. If $k \geq m/4$, then the fooling set $F_\pi^{\frac{m}{4},0}$ is well defined. If $k < m/4$, then the fooling set $F_\pi^{0,\frac{m}{4}}$ is well defined.*

Proof Let $I_{i_1}, I_{i_2}, ..., I_{i_k}$ be the intervals that are split by π. If $k \geq m/4$, then by Lemma 9, there exists pairwise inequivalent strings $a_1, ..., a_{m/4}, b_1, ..., b_{m/4}$ such that $a_j, b_j : I_{i_i} \to \{0, 1\}$ are strings with domain I_{i_j}, where both $\pi(a_j)|_{\pi(I_{i_j}) \cap L}$ and $\pi(a_j)|_{\pi(I_{i_j}) \cap R}$ are even, and both $\pi(b_j)|_{\pi(I_{i_j}) \cap L}$ and $\pi(b_j)|_{\pi(I_{i_j}) \cap R}$ are odd. Therefore, these strings can be used to define the fooling set $F_\pi^{\frac{m}{4},0}$.

On the other hand, if $k < m/4$, then Proposition 3 implies that there are at least $l = m/2 - 1 - k \geq m/4$ pairs $(\ell_1, r_1), (\ell_2, r_2), ..., (\ell_l, r_l)$ such that $\pi(I_{\ell_j}) \subseteq L$ and $\pi(I_{r_j}) \subseteq R$ for every $j \in \{1, ..., l\}$. Let $c_1, ..., c_{m/4}, d_1, ..., d_{m/4}$ be pairwise inequivalent even strings such that for each $j \in \{1, ..., m/4\}$, $c_j : I_{\ell_j} \to \{0, 1\}$ is a string with domain I_{ℓ_j}, and $d_j : I_{r_j} \to \{0, 1\}$ is a string with domain I_{r_j}. Note that these strings exist, since there are at least m even distinct strings with domain I_{ℓ_j} and m even distinct strings with domain I_{r_j} for each $j \in \{1, ..., k\}$. Therefore, these strings can be used to construct the fooling set $F^{0,m/4}$. □

5 Super-Linear Lower Bounds for Read-Once Circuits Excluding a Minor

In this section we show that exponential lower bounds for the symmetric non-deterministic complexity of a function $f_n : \{0,1\}^n \to \{0,1\}$ imply super-linear lower bounds for the size of read-once circuits excluding a fixed graph H as a minor. We start by establishing a connection between the symmetric-NSC of a circuit, and its pathwidth. More precisely, we show that any function that can be computed by a read-once circuit of pathwidth k has symmetric-NSC at most $2^k \cdot |\mathcal{C}|$.

Definition 6 (Path Decomposition) A path decomposition of a circuit $\mathcal{C} = (V, E, \mathbf{g})$ is a sequence $\mathcal{P} = (B_1, B_2, ..., B_m)$ of subsets of vertices of G satisfying the following properties.

i) $V = \bigcup_{i=1}^{n} B_i$.
ii) For each $i, j, k \in \mathbb{N}$ with $i < j < k$, $B_i \cap B_k \subset B_j$.
iii) For each edge $(u, v) \in E$ there is an j such that $\{u, v\} \subseteq B_j$.

The sets B_i are the bags of the decomposition. The path-width of $\mathcal{P}$ is defined as the size of its largest bag minus one. In other words $\boldsymbol{pw}(G, \mathcal{P}) = \max_i\{|B_i|\} - 1$. The pathwidth of a graph G is defined as $\boldsymbol{pw}(G) = \min_{\mathcal{P}} \boldsymbol{pw}(G, \mathcal{P})$ where $\mathcal{P}$ ranges over all path decompositions of G. Let $\mathcal{C}$ be a read-once circuit and $\mathcal{P} = (B_1, B_2, ..., B_m)$ be a path decomposition of $\mathcal{C}$. If v is a vertex of $\mathcal{C}$ then we let $\mathit{first}(v, \mathcal{P})$ denote the smallest i such that $v \in B_i$. We say that a path decomposition $\mathcal{P}$ is *ordered* if for each two vertices v and v', we have that $\mathit{first}(v, \mathcal{P}) \neq \mathit{first}(v', \mathcal{P})$. It can be shown that if a circuit $\mathcal{C}$ has m vertices and pathwidth w, then $\mathcal{C}$ has an *ordered* path decomposition with $O(m)$ bags and width $O(w)$. If $\mathcal{P}$ is an ordered path decomposition of a read-once circuit $\mathcal{C}$ and x is a variable labeling input v, then we define $\mathit{first}(x, \mathcal{P}) = \mathit{first}(v, \mathcal{P})$.

Theorem 8 *Let $\mathcal{C}$ be a read-once circuit and $\mathcal{P} = (B_1, B_2, ..., B_m)$ be a good path decomposition of $\mathcal{C}$ of width w. Let $x_1 x_2 ... x_n$ be an ordering of the variables of $\mathcal{C}$ such that $\mathit{first}(x_i, \mathcal{P}) < \mathit{first}(x_{i+1}, \mathcal{P})$ for each $i \in \{1, ..., n-1\}$. Then for each $b \in \Sigma$, one can construct an NFA with $m \cdot |\Sigma|^{O(w)}$ states accepting the following language.*

$$\mathcal{L}(\mathcal{C}, b) = \{a_1 a_2 ... a_n \in \Sigma^n \mid C(a_1 a_2 ... a_n) = b\}.$$

Theorem 8 will be proven Section 5.1. As a corollary of Theorems 8 and 4 we have a trade-off between the size of a circuit and its pathwidth.

Theorem 9 *Let $f_n : \{0,1\}^n \to \{0,1\}$ be a function of symmetric-NSC $\mathit{snsc}(f_n)$. Then for any read-once circuit computing f_n, the following inequality is satisfied.*

$$\boldsymbol{pw}(\mathcal{C}) + \log |\mathcal{C}| \geq \log \mathit{snsc}(f_n). \tag{9}$$

It is well known that the pathwidth of any graph is greater than its treewidth by at most a multiplicative logarithmic factor [5]. In other words, the following relation

between the pathwidth and treewidth of a circuit (graph) can be verified: $\boldsymbol{pw}(\mathcal{C}) \leq \boldsymbol{tw}(\mathcal{C}) \cdot O(\log |\mathcal{C}|)$. Therefore, stated in terms of treewidth, (9) can be rewritten as follows.

$$\boldsymbol{tw}(\mathcal{C}) \cdot \log |\mathcal{C}| + \log |\mathcal{C}| \geq \Omega(\log snsc(f_n)). \tag{10}$$

Theorem 10 ([3], Corollary 24 of [16]) *For any fixed graph H, every H-minor-free graph G with s vertices has treewidth at most $O(\sqrt{s})$.*

Therefore, combining (10) with Theorem 10 we have the following theorem. We say that a circuit $\mathcal{C}$ is H-minor-free if its underlying undirected graph is H-minor-free.

Theorem 11 *Let $\mathcal{C}$ be an H-minor-free, read-once circuit computing a Boolean function $f : \{0,1\}^n \to \{0,1\}$. Then $|\mathcal{C}| \geq \Omega\left(\left(\frac{\log snsc(f_n)}{\log n}\right)^2\right)$.*

Finally, as a corollary of Theorems 11, 3 and 4 we have that the triangle-freeness function Δ_n, the triangle-parity function $\bigoplus \text{Clique}_{3,n}$ and the even element distinctness function τ_n require H-minor-free read-once circuits of near quadratic size.

Corollary 1 *Let $\mathcal{C}$, $\mathcal{C}'$ and $\mathcal{C}''$ be H-minor-free, read-once circuits computing the triangle-freeness function Δ_n, the triangle-parity function $\bigoplus \text{Clique}_{3,n}$ and the even element distinctness function τ_n respectively. Then $|\mathcal{C}| \geq \Omega(\frac{n^2}{\log^2 n})$, $|\mathcal{C}'| \geq \Omega(\frac{n^2}{\log^2 n})$, and $|\mathcal{C}''| \geq \Omega(\frac{n^2}{\log^4 n})$.*

5.1 Proof of Theorem 8

Let $\mathcal{C}$ be a read-once circuit and $\mathcal{P} = (B_1, B_2, ..., B_m)$ be a path decomposition of $\mathcal{C}$. If v is a vertex of $\mathcal{C}$ then we let $first(v, \mathcal{P})$ denote the smallest i such that $v \in B_i$. Analogously, we let $last(v, \mathcal{P})$ denote the largest i such that $v \in B_i$. If (u, v) is an edge in $\mathcal{C}$ then we let $first(u, v, \mathcal{P})$ denote the smallest i such that $\{u, v\} \subseteq B_i$. For each $i \in \{1, ..., m\}$, and each vertex $v \in B_i$ the set of active in-neighbors of v on B_i is defined as follows.

$$AI(i, v) = \{u \in B_i \mid (u, v) \in E \text{ and } first(u, v, \mathcal{P}) = i\}$$

Intuitively, if u is an in-neighbor of v, then u belongs to $AI(i, v)$ if B_i is the first bag containing both u and v together. We note that for each edge $(u, v) \in E$ there exists a unique i such that $u \in AI(i, v)$. Intuitively, given an initialization for the input variables of $\mathcal{C}$, we will simulate the evaluation of the gates of $\mathcal{C}$ directly into the path decomposition $\mathcal{P} = (B_1, B_2, ..., B_m)$. The value of a vertex v is computed gradually while the path decomposition is traversed from left to right. For each i, the set $AI(i, v)$ consists of those vertices in B_i whose value will be taken into consideration toward the computation of the total value of v. This process is well defined due to the fact that for each in-neighbor u of v, there exists precisely one i for which $u \in AI(i, v)$.

The process of propagating the value of a circuit along a path decomposition can be formalized via the notion of *evaluation* defined below (Definition 7). If v is a vertex of $\mathcal{C}$ labeled with a $\oplus$-gate of arity k and $AI(i, v) = \{u_1, ..., u_r\}$, then we write $\sum_{u \in AI(i,v)} t_i(u)$ as a shortcut for $t_i(u_1) \oplus t_i(u_2) \oplus ... \oplus t_i(u_r)$. Note that since the operation $\oplus$ is assumed to be associative and commutative, we do not need to worry with brackets nor with the order in which operations are performed. Also, for simplicity, we may assume that the operation $\oplus$ has an identity[5] so that we can set $\sum_{u \in AI(i,v)} t_i(u)$ as the identity of $\oplus$ whenever $AI(i, v)$ is empty.

Definition 7 (Evaluation) Let $\mathcal{C} = (V, E, \mathbf{g})$ be a circuit where all inputs are initialized, and let $\mathcal{P} = (B_1, B_2, ..., B_m)$ be a path decomposition of $\mathcal{C}$. An evaluation of $\mathcal{C}$ on $\mathcal{P}$ is a sequence $\mathcal{E} = (p_1, t_1), (p_2, t_2), ..., (p_n, t_m)$ of pairs of functions $p_i : B_i \to \Sigma$ and $t_i : B_i \to \Sigma$ such that the following conditions are satisfied.

1. For every $i \in \{1, ..., m\}$, and every $v \in B_i$,
   (a) if v is an initialized input vertex then $p_i(v) = t_i(v) = \mathbf{g}(v)$,
   (b) if $i = first(v, \mathcal{P})$ then $p_i(v) = \sum_{u \in AI(i,v)} t_i(u)$.
   (c) If $i = last(v, \mathcal{P})$ then $p_i(v) = t_i(v)$.
2. For every $i \in \{1, ..., m - 1\}$, and every $v \in B_i \cap B_{i+1}$,
   (a) $t_{i+1}(v) = t_i(v)$.
   (b) $p_{i+1}(v) = p_i(v) + \sum_{u \in AI(i,v)} t_{i+1}(u)$.

For each i, and each vertex $v \in B_i$, we say that $t_i(v)$ is the total evaluation of v on bag B_i, while $p_i(v)$ is the partial evaluation of v on B_i. Intuitively the value of $t_i(v)$ is preemptively guessed, and $t_i(v) = t_j(v)$ for each two bags B_i and B_j containing v. On the other hand, the partial evaluation $p_i(v)$ keeps track of the partial sum corresponding to all inputs of v that have appeared up to bag B_i. At the last bag in which v occurs, the partial evaluation of v must equal its total evaluation.

To formalize the intuitive explanation given above, let $\mathcal{E}$ be an evaluation of $\mathcal{C}$ on $\mathcal{P}$ and let $I_{\mathcal{E}} : Inputs(\mathcal{C}) \to \Sigma$ denote the function that initializes the inputs of $\mathcal{C}$ consistently with the values assigned by $\mathcal{E}$ to these inputs along the path decomposition. More precisely, for each input u of C, $I_{\mathcal{E}}(u) = a$ if and only if there is some i such that $u \in B_i$ and $t_i(u) = a$. We denote by $\text{val}(C, I_{\mathcal{E}}, v)$ the value of the circuit at gate v when the inputs of $\mathcal{C}$ are initialized according to $I_{\mathcal{E}}$. Lemma 10 below states that an evaluation $\mathcal{E}$ of a read-once circuit $\mathcal{C}$ on a path decomposition $\mathcal{P}$ is consistent with the computation realized by $\mathcal{C}$ when its inputs are initialized according to $I_{\mathcal{E}}$. In other words, $\text{val}(\mathcal{C}, I_{\mathcal{E}}, v) = t_i(v)$ for each i and each $v \in B_i$.

Lemma 10 *Let $\mathcal{C}$ be a read-once circuit, $\mathcal{P} = (B_1, B_2, ..., B_m)$ be a path decomposition of $\mathcal{C}$ and $\mathcal{E} = (p_1, t_1)...(p_m, t_m)$ be an evaluation of $\mathcal{C}$ on $\mathcal{P}$. Then for each $i \in \{1, ..., m\}$, and each vertex $v \in B_i$, $t_i(v) = \text{val}(\mathcal{C}, I_{\mathcal{E}}, v)$.*

[5] For instance, the identity of $\wedge$ is 1, while the identity of $\vee$ is 0. The requirement of an identity can easily be removed by replacing Condition 1b with a slightly more complicated initialization condition.

Proof The proof is by induction on the depth of the vertex v in $\mathcal{C}$. In the base case, v is an input vertex. In this case, there exists some i for which $v \in B_i$ and $I_{\mathcal{E}}(v) = \text{val}(\mathcal{C}, I_{\mathcal{E}}, v) = t_i(v)$. Since the total sum $t_i(v)$ has the same value for each bag containing the vertex v, we have that $t_j(v) = I_{\mathcal{E}}(v)$. Now assume that the claim is valid for every vertex of depth at most d and let v be a vertex of depth $d+1$. Note that for each edge (u, v) there is a unique j such that $u \in AI(j, v)$. Therefore, by Condition 1b and Condition 2b, we have that if $i = \mathit{last}(v, \mathcal{P})$, then $p_i(v) = \sum_{(u,v)\in E} t(u) = \sum_{(u,v)\in E} \text{val}(\mathcal{C}, I_{\mathcal{E}}, u) = \text{val}(\mathcal{C}, I_{\mathcal{E}}, v)$. Since by Condition 1c, $p_i(v) = t_i(v)$, we have that $t_i(v) = \text{val}(\mathcal{C}, I_{\mathcal{E}}, v)$. □

Let $\mathcal{C}$ be a read-once circuit. For each variable x labeling a vertex v of $\mathcal{C}$, we write $\mathit{first}(x, \mathcal{P})$ in place of $\mathit{first}(v, \mathcal{P})$. In other words, $\mathit{first}(x, \mathcal{P})$ denotes the index of the first bag of $\mathcal{P}$ in which the vertex v occurs. Note that $\mathit{first}(x, \mathcal{P})$ is well defined, since in a read-once circuit each input variable labels at most one vertex. Now let $x_1, ..., x_n$ be the sequence of all variables labeling inputs of $\mathcal{C}$, such that $\mathit{first}(x_j, \mathcal{P}) < \mathit{first}(x_{j+1}, \mathcal{P})$ for each $j \in \{1, ..., n\}$. Then an assignment to the inputs of $\mathcal{C}$ can be represented by a string $a = a_1 a_2 ... a_n$ where for each $j \in \{1, ..., n\}$, a_j is the value assigned to variable x_j.

Proof of Theorem 8 For each $i \in \{1, ..., m\}$, we say that a pair of functions $p, t : B_i \to \Sigma$ is an assignment for B_i, if together, p and t satisfy Conditions 1a–1c of Definition 7. In our first step toward the proof of Theorem 8, we will construct an NFA that accepts all sequences of assignments encoding an evaluation of $\mathcal{C}$ on $\mathcal{P}$. To make this notion precise, for each $i \in \{1, ..., m\}$, let

$$\Gamma_i = \{(p, t) \mid (p, t) \text{ is an assignment for } B_i\}$$

be the set of all assignments for B_i. Since the width of $\mathcal{P}$ is w, we have that $|\Gamma_i| \leq |\Sigma|^{2w}$. We say that Γ_i is the alphabet of assignments of B_i. Now let $\Gamma = \bigcup_{i=1}^{m} \Gamma_i$. We construct an NFA $\mathcal{A} = (Q, \Gamma, \Delta, Q_0, F)$ that accepts a string $\mathcal{E} = (p_1, t_1)(p_2, t_2)...(p_m, t_m)$ over Γ if and only if $\mathcal{E}$ is an evaluation of $\mathcal{C}$ on $\mathcal{P}$. The set of states $Q = Q_0 \dot\cup Q_1 \dot\cup ... \dot\cup Q_m$ is partitioned into $m+1$ levels. The level $Q_0 = \{q_0\}$ has a unique initial state. For each $i \in \{1, ..., m\}$, the level $Q_i = \{q_{i,p,t} \mid (p, t) \in \Gamma_i\}$ has one state $q_{i,p,t}$ for each assignment (p, t) of the bag B_i. The final states of $\mathcal{A}$ are those lying at Q_m, that is to say, $F = Q_m$. We say that an assignment (p_i, t_i) of B_i is compatible with an assignment (p_{i+1}, t_{i+1}) of B_{i+1} if the functions p_i, p_{i+1}, t_i and t_{i+1} satisfy Conditions 2b and 2b of Definition 7. The transition relation Δ has one transition $(q_0, (p_1, t_1), q_{1,p_1,t_1})$ for each assignment (p_1, t_1) of the bag B_1, and one transition $(q_{i,p_i,t_i}, (p_{i+1}, t_{i+1}), q_{i+1,p_{i+1},t_{i+1}})$ for each pair (p_i, t_i) and (p_{i+1}, t_{i+1}) of compatible assignments. It is straightforward to check that $\mathcal{A}$ accepts a string $\mathcal{E} = (p_1, t_1)(p_2, t_2)...(p_m, t_m)$ if and only if for each $i \in \{1, ..., m\}$, (p_i, t_i) is an assignment of B_i, and for each $i \in \{1, ..., m-1\}$, (p_i, t_i) is compatible with (p_{i+1}, t_{i+1}). In other words, $\mathcal{E}$ is accepted by $\mathcal{A}$ if and only if $\mathcal{E}$ is an evaluation of $\mathcal{C}$ on $\mathcal{P}$. Now, let v be the output vertex of $\mathcal{C}$. Let $\mathcal{A}'$ be the automaton over Γ obtained by deleting all states $q_{i,p,t}$ such that $v \in B_i$ and $t(v) \neq b$. Then we have that $\mathcal{A}'$ accepts an evaluation $\mathcal{E}$ if and only if it causes the circuit $\mathcal{C}$ to evaluate to b.

Finally, we will construct an automaton $\mathcal{A}''$ that extracts from each such evaluation $\mathcal{E}$, only the information concerning the assignments of values to the uninitialized variables. Let $i_1 = first(x_1, \mathcal{P})$, ..., $i_n = first(x_n, \mathcal{P})$ and let v_i be the unique input vertex of $\mathcal{C}$ labeled with x_i. Consider the projection $\alpha : \Gamma \to \{0, 1, \varepsilon\}$ (where ε is the empty string) whose action is defined as follows.

$$\alpha(p_i, t_i) = \begin{cases} \varepsilon & \text{if } i \notin \{i_1, ..., i_n\} \\ 0 & \text{if } i \text{ in } \{i_1, ..., i_n\} \text{ and } t_i(v_i) = 0 \\ 1 & \text{if } i \text{ in } \{i_1, ..., i_n\} \text{ and } t_i(v_i) = 1 \end{cases} \tag{11}$$

One can then construct in time $O(|Q| + |\Delta|)$ a non-deterministic finite automaton $\alpha(\mathcal{A})$ which accepts precisely those strings $\alpha(p_1, t_1)\alpha(p_2, t_2)...\alpha(p_m, t_m) \in (\Sigma \cup \{\varepsilon\})^*$ with the property that $(p_1, t_1)(p_2, t_2)...(p_m, t_m)$ is accepted by $\mathcal{A}$. If we regard the symbol ε as being the empty string symbol, then the automaton $\alpha(\mathcal{A})$ may be regarded as an automaton with ε-transitions that accepts a string $a_1a_2...a_n \in \Sigma^n$ if and only if the circuit $\mathcal{C}$ evaluates to b when its inputs $x_1x_2...x_n$ are are initialized with $a_1a_2...a_n$. This automaton can be transformed into an automaton without ε-transitions with the same number of states by folklore results in automata theory. □

Acknowledgements The author would like to thank Pavel Pudlák for interesting discussions on circuit lower-bounds, Pavel Hrubeš for pointing me out to reference [22], Michal Koucký and Bruno Loff for useful feedback during a seminar presentation of this work, and anonymous referees for valuable comments.

References

1. Alekhnovich, M., Razborov, A.A.: Satisfiability, branch-width and tseitin tautologies. In: Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS 2002), pp. 593–603 (2002)
2. Allender, E., Chen, S., Lou, T., Papakonstantinou, P.A., Tang, B.: Width-parametrized SAT: time–space tradeoffs. Theory of Computing **10**(12), 297–339 (2014)
3. Alon, N., Seymour, P., Thomas, R.: A separator theorem for graphs with an excluded minor and its applications. In: Proceedings of the 22nd Symposium on Theory of Computing (STOC 1990), pp. 293–299. ACM (1990)
4. Blum, N.: A boolean function requiring 3n network size. Theor. Comput. Sci. **28**(3), 337–345 (1983)
5. Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theor. Comput. Sci. **209**(1–2), 1–45 (1998)
6. Broering, E., Lokam, S.V.: Width-based algorithms for SAT and CIRCUIT-SAT. In: Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT 2004), Volume 2919 of LNCS, pp. 162–171. Springer (2003)
7. Calabro, C.: A lower bound on the size of series-parallel graphs dense in long paths. Electronic Colloquium on Computational Complexity (ECCC), 15(110) (2008)
8. de Oliveira Oliveira, M.: Size-treewidth tradeoffs for circuits computing the element distinctness function. In: Proceedings of the 33Rd Symposium on Theoretical Aspects of Computer Science (STACS 2016), Volume 47 of LIPIcs, pp. 56:1–56:14 (2016)
9. Demenkov, E., Kulikov, A.S.: An elementary proof of a $3n - o(n)$ lower bound on the circuit complexity of affine dispersers. In: Proceedings of the 36th International Symposium on Mathematical Foundations of Computer Science (MFCS 2011), Volume 6907 of LNCS, pp. 256–265. Springer (2011)
10. Diestel, R.: Graph Theory, volume 173 of Graduate Texts in Mathematics. Springer (2000)
11. Duris, P., Hromkovic, J., Jukna, S., Sauerhoff, M., Schnitger, G.: On multi-partition communication complexity. Inf. Comput. **194**(1), 49–75 (2004)

12. Find, M.G., Golovnev, A., Hirsch, E.A., Kulikov, A.S.: A better-than-3n lower bound for the circuit complexity of an explicit function. In: Proceedings of the 57th Annual Symposium on Foundations of Computer Science (FOCS 2016), pp. 89–98. IEEE (2016)
13. Gál, A., Jang, J.-T.: A generalization of Spira's theorem and circuits with small segregators or separators. In: Proceedings of the 38th Conference on Theory and Practice of Computer Science (SOFSEM 2012), Volume 7147 of LNCS, pp. 264–276. Springer (2012)
14. Georgiou, K., Papakonstantinou, P.A.: Complexity and algorithms for well-structured k-sat instances. In: Proceedings of the 11th International Conference on Theory and Applications of Satisfiability Testing (SAT 2008), Volume 4996 of LNCS, pp. 105–118. Springer (2008)
15. Glaister, I., Shallit, J.: A lower bound technique for the size of nondeterministic finite automata. Inf. Process. Lett. **59**(2), 75–77 (1996)
16. Grohe, M.: Local tree-width, excluded minors, and approximation algorithms. Combinatorica **23**(4), 613–632 (2003)
17. Håstad, J.: The shrinkage exponent of de morgan formulas is 2. SIAM J. Comput. **27**(1), 48–64 (1998)
18. He, J., Liang, H., Sarma, J.M.: Limiting negations in bounded treewidth and upward planar circuits. In: In Proceedings 35th International Symposium on Mathematical Foundations of Computer Science (MFCS 2010), Volume 6281 of LNCS, pp. 417–428. Springer (2010)
19. Hromkovič, J.: Communication complexity and lower bounds on multilective computations. RAIRO - Theoretical Informatics and Applications **33**(02), 193–212 (1999)
20. Iwama, K., Morizumi, H.: An explicit lower bound of 5n-o(n) for boolean circuits. In: Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science (MFCS 2002), Volume 2420 of LNCS, pp. 353–364. Springer (2002)
21. Jansen, M.J., Sarma, J.: Balancing bounded treewidth circuits. Theory Comput. Syst. **54**(2), 318–336 (2014)
22. Jukna, S.: Boolean Function Complexity: Advances and Frontiers, volume 27 of Algorithms and combinatorics. Springer (2012)
23. Lachish, O., Raz, R.: Explicit lower bound of 4.5 n-o(n) for boolena circuits. In: Proceedings of the 33rd Annual Symposium on Theory of Computing (STOC 2001), pp. 399–408. ACM (2001)
24. Lipton, R.J., Tarjan, R.E.: Applications of a planar separator theorem. SIAM J. Comput. **9**(3), 615–627 (1980)
25. Markov, I.L., Shi, Y.: Constant-degree graph expansions that preserve treewidth. Algorithmica **59**(4), 461–470 (2011)
26. Nestoridis, N.V., Thilikos, D.M.: Square roots of minor closed graph classes. Discrete Appl. Math. **168**, 34–39 (2014)
27. Nečiporuk: On a Boolean function. Soviet Math. Dokl. **7**(4), 999–1000 (1966)
28. Papadimitriou, C.H., Sipser, M.: Communication complexity. J. Comput. Syst. Sci. **28**(2), 260–269 (1984)
29. Paturi, R., Pudlák, P.: Circuit lower bounds and linear codes. J. Math. Sci. **134**(5), 2425–2434 (2006)
30. Robertson, N., Seymour, P.D.: Graph minors. XIII. The disjoint paths problem. Journal of Combinatorial Theory, Series B **63**(1), 65–110 (1995)
31. Santhanam, R., Srinivasan, S.: On the limits of sparsification. In: Proceedings of 39th the International Conference on Automata, Languages, and Programming (ICALP 2012), Volume 7391 of LNCS, pp. 774–785. Springer (2012)
32. Savage, J.E.: Planar circuit complexity and the performance of VLSI algorithms. In: INRIA Report 77 (1981). Also in VLSI Systems and Computations, pp. 61–67. Computer Science Press, Rockwille, MD (1981)
33. Turán, G.: On the complexity of planar boolean circuits. Comput. Complex. **5**(1), 24–42 (1995)
34. Valiant, L.G.: Graph-theoretic arguments in low-level complexity. In: Proceedings of the 6th Symposium on Mathematical Foundations of Computer Science, Volume 53 of LNCS, pp. 162–176 (1977)
35. Wegener, I.: Branching Programs and Binary Decision Diagrams: Theory and Applications. SIAM (2000)