On the read-once property of branching programs and CNFs of bounded treewidth

Igor Razgon

Department of Computer Science and Information Systems, Birkbeck, University of London
igor@dcs.bbk.ac.uk

Abstract. In this paper, given a parameter $k$, we demonstrate an infinite class of CNFs of treewidth at most $k$ of their primary graphs such that the equivalent acyclic (syntactic) read-once switching and rectifier networks (AROSRN$s$) are of size at least $n^{ck}$ for some universal constant $c$. Thus we rule out the possibility of fixed-parameter space complexity of AROSRNs parameterized by the smallest treewidth of the equivalent CNF. The proposed result is a significant enhancement of the lower bound for Ordered Binary Decision Diagrams (OBDD$s$) proved in [13]. We use the above lower bound to obtain a quasi-polynomial separation between Free Binary Decision Diagrams and Decision Decomposable Negation Normal Forms, essentially matching the simulation result of [1] and thus proving the tightness of the latter.

1 Introduction

Read-once Branching Programs (ROBPs) is a well known representation of Boolean functions. Oblivious ROBPs, better known as Ordered Binary Decision Diagrams (OBDD$s$), is a subclass of ROBPs, very well known because of its applications in the area of verification [2]. An important procedure in these applications is transformation of a CNF into an equivalent OBDD. The resulting OBDD can be exponentially larger than the initial CNF, however a space efficient transformation is possible for special classes of functions. For example, it has been shown in [6] that a CNF of treewidth $k$ of its primal graph can be transformed into an OBDD of size $O(n^k)$. A natural question is if the upper bound can be made fixed-parameter i.e. of the form $f(k)n^c$ for some constant $c$. In [13] we showed that it is impossible by demonstrating that for each sufficiently large $k$ there is an infinite class of CNFs of treewidth at most $k$ whose smallest OBDD is of size at least $n^{k/5}$.

In this paper we report a follow up result showing that essentially the same lower bound holds for Acyclic Read-Once Switching and Rectifier Networks (AROSRN$s$) [1]. In particular we show that there is a constant $0 < c < 1$ such that for each sufficiently large $k$ there is an infinite class of CNFs of treewidth at most $k$ (of their primal graphs) for which the space complexity of the equivalent AROSRNs is at least $n^{ck}$.

1 Throughout this paper, we assume the read-once property to be syntactic, that is applied to all root-leaf paths of the considered branching programs. See Section 2 for the exact definitions.
This result is a significant enhancement of the result of [13]. Indeed, OBDDs are a subclass ROBPs and there is exponential separation between the classes (that is, there is a family of functions that can be represented by poly-size ROBPs but require exponential size OBDDs). Further on, a ROBP is a special case of Nondeterministic ROBP (NROBP) and there is an exponential separation between ROBPs and NROBPs ([16], Corollary 10.2.3). Finally on, a NROBP can be seen as a special case of an AROSRN. Thus the proposed result shows that read-once branching programs are inherently incapable to efficiently compute CNFs of bounded treewidth.

We also demonstrate that the proposed result can be used in the non-parameterized context. In particular, using this result, we provide a quasi-polynomial separation between AROSRNs and a subset of decomposable negation normal forms (DNNFs) [3] known as decision DNNF. More precisely, we demonstrate a family of CNFs that can be expressed as decision DNNFs of size $O(n^2)$ but the space complexity of AROSRNs is $n^{\Omega(\log n)}$. The motivation for this result is described below.

DNNF is a representation of Boolean functions well known in the areas of knowledge representation and databases. DNNFs are much more succinct than ROBPs. In fact a ROBP can be seen as a special case of DNNF [5] and there is an exponential separation between these two representations. Like in the case of OBDDs, transformation from a CNF to an equivalent DNNF is an important operation in the related applications. One remarkable property of DNNFs is their FPT space complexity on CNFs of bounded treewidth. In particular, a CNF of treewidth $k$ can be transformed into a DNNF of size $O(2^k n)$. In fact this property is preserved for a number of restricted DNNF subclasses, one of them is known as decision DNNF [10]. Interestingly, the possibility of exponential separation from ROBP is not preserved for decision DNNFs: it has been shown in [1] that a decision DNNF of size $N$ can be simulated by a ROBP of size $O(N \log N)$. Our result shows that this upper bound is essentially tight. Indeed, since ROBP can essentially be seen as a special case of AROSRN, this result implies quasi-polynomial separation between ROBP and decision DNNF, essentially matching the upper bound of [1].

To prove the proposed parameterized lower bound, we use monotone 2-CNFs (their clauses are of form $(x_1 \lor x_2)$ where $x_1$ and $x_2$ are 2 distinct variables). These CNFs are in one-to-one correspondence with graphs having no isolated vertices: variables correspond to vertices and 2 variables occur in the same clause if and only if the corresponding vertices are adjacent. This correspondence allows us to use these CNFs and graphs interchangeably. We introduce the notion of Matching Width (MW) of a graph $G$ and prove two theorems. One of them states that an AROSRN equivalent to a monotone 2-CNF with the corresponding graph $G$ having MW at least $t$ is of size at least $2^{t/a}$ where $a$ is a constant dependent on the max-degree of $G$. The second theorem states that for each sufficiently large $k$ there is an infinite family of graphs of treewidth $k$ and max-degree 5 whose MW is at least $b \cdot \log n \cdot k$ for some constant $b$ independent of $k$. The main theorem immediately follows from replacement of $t$ in the former lower bound by the latter one.

The strategy outlined above is similar to that we used in [13]. However, there are two essential differences. First, due to a much more ‘elusive’ nature of AROSRNs compared to that of OBDD, the counting argument is more sophisticated and more restrictive: it
applies only to CNFs whose graphs are of constant degree. Due to this latter aspect, the target set of CNF instances requires a more delicate construction and reasoning.

We believe the proposed parameterized lower bound is interesting from the parameterized complexity theory perspective because it contributes to the understanding of (concrete) parameterized space complexity of various representations of Boolean functions. We see at least two reasons why this research direction is worth to explore. First, the results of of this kind are closely related (through substitution of the parameters with appropriate functions of $n$) to the classical, non-parameterized complexity of Boolean function. For example, the famous result of Razborov providing the first non-polynomial lower bound for the space complexity of monotone circuits can be seen formulated in the parameterized setting as a space $n^{\sqrt{k}}$ lower bound for monotone circuits testing whether the given graph has a clique of size $k$ [8].

The second reason why we believe that the parameterized complexity of Boolean functions is an interesting research direction is that parameterized upper bounds on the space complexity of Boolean functions are important in applications related to verification, knowledge representation, and databases. In fact, quite a few such upper bounds are already known (e.g. [3,4,10,9,7]). Therefore, it is interesting to see if advanced parameterized complexity methodologies can be applied in order to enhance these upper bounds and to obtain new ones.

It is also worth noting that the treewidth is a very important parameter in the context of Boolean functions. Besides the fact that it serves as the parameter for most of the above upper bounds, it also has some interesting and surprising properties. Indeed, let us define the treewidth ($twd$) and cliquewidth ($cwd$) of a Boolean function as the smallest treewidth (resp. cliquewidth) of a Boolean circuit (over the standard $\lor, \land, \neg$-basis) implementing this function. Then it can be shown [14] that $cwd \leq 18 \ast twd$ in contrast to graphs where there are classes of constant cliquewidth and unbounded treewidth.

The rest of the paper is organized as follows. Section 2 introduces the necessary background. Section 3, 4, and 5 prove the parameterized lower bound (the last two sections prove auxiliary theorems used for the lower bound proof in section 3). Section 6 establishes the quasipolynomial separation between decisionDNNF and AROSRN. Section A of the appendix shows that AROSRNs can be simulated by uniform AROSRN with only a linear increase in space. Since the lower bound proved in Sections 3-5 is proved for uniform AROSRNs, this simulation shows that this lower bound holds for general AROSRN.

2 Preliminaries

In this paper by a set of literals we mean one that does not contain an occurrence of a variable and its negation. For a set $S$ of literals we denote by $Var(S)$ the set of variables whose literals occur in $S$. If $F$ is a Boolean function or its representation by a specified structure, we denote by $Var(F)$ the set of variables of $F$. A truth assignment to $Var(F)$ on which $F$ is true is called a satisfying assignment of $F$. A set $S$ of literals represents the truth assignment to $Var(S)$ where variables occurring positively in $S$ (i.e. whose literals in $S$ are positive) are assigned with true and the variables occurring negatively are assigned with false. We denote by $F_S$ a function whose set of satisfying
assignments consists of \(S'\) such that \(S \cup S'\) is a satisfying assignment of \(F\). We call \(F_S\) a subfunction of \(F\).

Let us define the Boolean functions representations considered in this paper.

The central notion considered in this paper is an acyclic read-once switching-and-rectifier network (AROSRN). An AROSRN \(Y\) implementing (computing) a function \(F\) is a directed acyclic graph (DAG) (with possible multiple edges) with one leaf, one root, and with some edges labelled by literals of the variables of \(F\) in a way that there is no directed path having two edges labelled with literals of the same variable. We denote by \(A(P)\) the set of literals labelling edges of a directed path \(P\) of \(Y\).

The connection between \(Y\) and \(F\) is defined as follows. Let \(P\) be a path from the root to the leaf of \(Y\). Then any extension of \(A(P)\) to the truth assignment of all the variables of \(F\) is a satisfying assignment of \(F\). Conversely, let \(A\) be a satisfying assignment of \(F\). Then there is a path \(P\) from the root to the leaf of \(Y\) such that \(A(P) \subseteq A\).

The notion of AROSRN can be easily seen as a generalization of Non-deterministic Read Once Branching Program (NROBP). The latter is defined as a DAG \(Z\) with one root and two leaves. Some of non-leaf nodes are labelled with variables so that no variable occurs as a label twice on a directed path of \(Z\). A node labelled with a variable has two outgoing edges one labelled with \(true\) the other with \(false\). Finally, the leaves are labelled with \(true\) and \(false\).

It is convenient to see each edge \(e\) labelled with \(true\) or \(false\) being in fact respectively labelled with the positive or negative literal of the variable labelling the tail of \(e\). With such a labelling an assignment \(A(P)\) associated with each directed path of \(Z\) is simply the set of literals labelling the edges of \(P\). The satisfying assignments of the function computed by \(Z\) are precisely those that are extensions of \(A(P)\) for paths \(P\) from the root to the \(true\) leaf.

It is not hard to see that for any function that is not constant \(false\), NROBP can be thought as a special case of AROSRN. Indeed, with edges labelled by literals as specified in the previous paragraph, remove the labels from the vertices, remove the \(false\) leaf as well as all nodes of \(Z\) from which the \(true\) leaf is not reached and the obtained graph is an AROSRN computing exactly the same function as \(Z\).

Remark. The nodes of a NROBP that are not marked by variables are called guessing nodes. The (deterministic) ROBP, a.k.a. Free Binary Decision Diagram is a NROBP without guessing nodes. If, in addition, there is a permutation \(SV\) of the variables of a ROBP \(Z\) such that \(x\) precedes \(y\) whenever \(Z\) has a path from a vertex labelled with \(x\) to a vertex labelled with \(y\) then \(Z\) is called an oblivious ROBP, a.k.a. Ordered Binary Decision Diagrams (OBDD).

We say that a AROSRN \(Y\) is uniform if the following is true. Let \(a\) be a node of \(Y\) and let \(P_1\) and \(P_2\) be 2 paths from the root of \(Y\) to \(a\). Then \(\text{Var}(A(P_1)) = \text{Var}(A(P_2))\). That is, these paths are labelled by literals of the same set of variables. Also, if \(P\) is a path from the root to the leaf of \(Y\) then \(\text{Var}(A(P)) = \text{Var}(F)\). Thus there is a one-to-one correspondence between the sets of literals labelling paths from the root to the leaf of \(Y\) and the satisfying assignments of \(F\).

All the AROSNRs considered in Sections 3-6 of this paper are uniform. This assumption does not affect our main result because an arbitrary AROSRN can be transformed into a uniform one at the price of \(O(n)\) times increase of the number of edges.
For the sake of completeness, we provide the transformation and its correctness proof in the appendix. We use the construction described in the proof sketch of Proposition 2.1 of [12].

Now we are going to define the Decomposable Negation Normal Form (DNNF) and its subclass decision DNNF for which we prove a separation result in Section 6.

Remark. The only thing we need to know for this separation result is that a CNF of a bounded primal graph treewidth can be transformed into an FPT-size decision DNNF [10]. That is, the two paragraphs below are not needed for the technical reasoning. We provide these definitions for the sake of completeness in the sense that all the representations of Boolean functions occurring in the statements of this paper are explicitly defined.

Recall that a Boolean circuit over the $\lor$, $\land$, $\neg$ is called de Morgan circuit if the negations are applied only to the input (variable) gates. Next, we define a decomposable node. Let $x$ be a gate of a Boolean circuit $X$. We denote by $V\text{Reach}(x)$ the set of variables such that $x$ is reachable from their respective input gates. We say that $x$ is decomposable if for any two in-neighbours $y_1$ and $y_2$ of $x$, $V\text{Reach}(y_1) \cap V\text{Reach}(y_2) = \emptyset$. A DNNF is a de-Morgan circuit with all the AND-nodes being decomposable.

We say that an OR-node $x$ of a DNNF is a decision node (see Figure 1) if it is binary, both its in-neighbours $y_1$ and $y_2$ are AND-nodes and there is a variable $x$ such that $x$ is an input of, say $y_1$ and $\neg x$ is an input of $y_2$. A DNNF is called decision DNNF if all its OR nodes are decision ones. See Figure 2 showing a DNNF and a decision DNNF for the same function. Note that for the latter we use both variable and constant input gates.

Given a graph $G$, its tree decomposition is a pair $(T, B)$ where $T$ is a tree and $B$ is a set of bags $B(t)$ corresponding to the vertices $t$ of $T$. Each $B(t)$ is a subset of $V(G)$ and the bags obey the rules of union (that is, $\bigcup_{t \in V(T)} B(t) = V(G)$), containment (that is, for each $\{u, v\} \in E(G)$ there is $t \in V(T)$ such that $\{u, v\} \subseteq B(t)$), and connectedness (that is for each $u \in V(G)$, the set of all $t$ such that $u \in B(t)$ induces a subtree of $T$). The width of $(T, B)$ is the size of the largest bag minus one. The treewidth of $G$ is the smallest width of a tree decomposition of $G$. 

Fig. 1. A decision node
Given a CNF \( \phi \), its primal graph has the set of vertices corresponding to the variables of \( \phi \). Two vertices are adjacent if and only if there is a clause of \( \phi \) where the corresponding variables both occur.

### 3 The parameterized lower bound

A monotone 2-CNFs has clauses of the form \((x \lor y)\) where \(x\) and \(y\) are two distinct variables. Such CNFs can be put in one-to-one correspondence with graphs that do not have isolated vertices. In particular, let \( G \) be such a graph. Then \( G \) corresponds to a 2CNF \( \phi(G) \) whose variables are the vertices of \( G \) and the set of clauses is \( \{(u \lor v) | \{u, v\} \in E(G)\} \). These notions, together with the corresponding AROSRN, are illustrated on Figure 3. It is not hard to see that \( G \) is the primal graph of \( \phi(G) \), hence we can refer to the treewidth of \( G \) as the primal graph treewidth of \( \phi(G) \).

The following theorem is the main result of this paper.

**Theorem 1.** There is a constant \( c \) such that for each \( k \geq 50 \) there is an infinite class \( G \) of graphs each of treewidth of at most \( k \) such that for each \( G \in G \), the smallest AROSRN equivalent to \( \phi(G) \) is of size at least \( n^{k/c} \), where \( n \) is the number of variables of \( \phi(G) \).

In order to prove Theorem 1, we introduce the notion of matching width (MW) of a graph and state two theorems proved in the subsequent two sections. One claims that if the max-degree of \( G \) is bounded then the size of a AROSRN realizing \( \phi(G) \) is exponential in the MW of \( G \). The other theorem claims that for each sufficiently large \( k \) there is an infinite class of graphs of bounded degree and of treewidth at most \( k \) whose MW is at least \( b \ast \log n \ast k \) for some universal constant \( b \). Theorem 1 will follow as an immediate corollary of these two theorems.

---

\(^2\) Notice that on the AROSRN in Figure 3, there is a path where \( v_2 \) occurs before \( v_3 \) and a path where \( v_3 \) occurs before \( v_2 \). Thus this AROSRN, although uniform, is not oblivious.
Definition 1. Matching width.

Let \( SV \) be a permutation of \( V(G) \) and let \( S_1 \) be a prefix of \( SV \) (i.e. all vertices of \( SV \setminus S_1 \) are ordered after \( S_1 \)). The matching width of \( S_1 \) is the size of the largest matching consisting of the edges between \( S_1 \) and \( V(G) \setminus S_1 \). The matching width of \( SV \) is the largest matching width of a prefix of \( SV \). The matching width of \( G \), denoted by \( mw(G) \), is the smallest matching width of a permutation of \( V(G) \).

Remark. The above definition of matching width is a special case of the notion of maximum matching width as defined in [15].

To illustrate the notion of matching width recall that \( C_n \) and \( K_n \) respectively denote a cycle and a complete graph of \( n \) vertices. Then, for a sufficiently large \( n \), \( mw(C_n) = 2 \). On the other hand \( mw(K_n) = \lceil n/2 \rceil \).

Theorem 2. For each integer \( i \) there is a constant \( a_i \) such that for any graph \( G \) the size of \( \text{AROSRN} \) realizing \( \phi(G) \) is at least \( 2^{mw(G)/a_x} \) where \( x \) is the max-degree of \( G \).

Theorem 3. There is a constant \( b \) such that for each \( k \geq 50 \) there is an infinite class \( G \) of graphs of degree at most \( 5 \) such that the treewidth of all the graphs of \( G \) is at most \( k \) and the matching width of each \( G \in G \) is at least \( (\log n + k)/b \) where \( n = |V(G)| \).

Now we are ready to prove Theorem 1.

Proof of Theorem 1. Let \( G \) be the class whose existence is claimed by Theorem 3. By Theorem 2, for each \( G \in G \) the size of \( \text{AROSRN} \) realizing \( \phi(G) \) is of size at least \( 2^{mw(G)/a_x} \). Further on, by Theorem 3 \( mw(G) \geq (\log n + k)/b \), for some constant \( b \). Substituting the inequality for \( mw(G) \) into the lower bound \( 2^{mw(G)/a_x} \) supplied by Theorem 2 we get that the size of a \( \text{AROSRN} \) is at least \( 2^{(\log n + k)/c} \) where \( c = a_5 + b \). Replacing \( 2^{\log n} \) by \( n \) gives us the desired lower bound.

From now on, the proof is split into two independent parts: Section 4 proves Theorem 2 and Section 5 proves Theorem 3.

We sometimes treat sequences as sets, the correct use will be always clear from the context.
4 Proof of Theorem

Recall that the vertices of graph $G$ serve as variables in $\phi(G)$. That is, in the truth assignments to $\text{Var}(\phi(G))$, the vertices are treated as literals and may occur positively or negatively. Similarly for a path $P$ of a arosrn $Z$ implementing $\phi(G)$, we say that a vertex $v \in V(G)$ occurs on $P$ if either $v$ and $\neg v$ labels an edge of $P$. In the former case this is a positive occurrence, in the latter case a negative one.

Recall that a Vertex Cover (VC) of $G$ is $V' \subseteq V(G)$ incident to all the edges of $E(G)$.

**Observation 1** $S$ is a satisfying assignment of $\phi(G)$ if and only if the vertices of $G$ occurring positively in $S$ form a VC of $G$. Equivalently, $V' \subseteq V(G)$ is the set of all vertices of $G$ occurring positively on a root-leaf path of $Z$ if and only if $V'$ is a VC of $G$.

In light of Observation 1 we denote the set of all vertices occurring positively on a root-leaf path $P$ of $Z$ by $VC(P)$.

The proof of Theorem 2 requires two intermediate statements. For the first statement, let $a$ be a node of an arosrn $Z$. For an integer $t > 0$, we call $a$ a $t$-node if there is a set $S(a)$ of size at least $t$ such that for each root-leaf path $P$ passing through $a$, $S(a) \subseteq VC(P)$. To demonstrate the notion of a $t$-node, consider the two nodes denoted by circles in Figure 3. They are 2-nodes for the given arosrn, the witnessing set for the left-hand node is $\{v_1, v_4\}$ and for the right-hand node is $\{v_2, v_3\}$.

**Lemma 1.** Suppose that the matching width of $G$ is at least $t$. Then $t$-nodes of $Z$ form a root-leaf cut.

**Proof.** We need to show that each root-leaf path $P$ passes through a $t$-node. Due to the uniformity of $Z$, (the vertices of $G$ corresponding to) the labels of $P$ are explored from the root to the leaf form a permutation $SV$ of $V(G)$. Let $SV'$ be a prefix of the permutation witnessing the matching width at least $t$. In other words, there is a matching $M = \{\{u_1, v_1\}, \ldots, \{u_t, v_t\}\}$ of $G$ such that all of $u_1, \ldots, u_t$ belong to $SV'$, while all of $v_1, \ldots, v_t$ belong to $SV \setminus SV'$. Let $u$ be the last vertex of $SV'$ and let $a$ be the head of the edge of $P$ whose label is a literal of $u$. We claim that $a$ is a $t$-node with a witnessing set $S(a) = \{x_1, \ldots, x_t\}$ such that $x_i \in \{u_i, v_i\}$ for each $x_i$.

Indeed, observe that for each $\{u_i, v_i\}$ there is $x_i \in \{u_i, v_i\}$ such that $x_i \in VC(P)$ for each root-leaf path $P$ passing through $a$. Clearly for any root-leaf path $Q$ of $Z$, either $u_i \in VC(Q)$ or $v_i \in VC(Q)$ for otherwise $VC(Q)$ is not a vc of $G$ in contradiction to Observation 1. Thus if such $x_i$ does not exist then there are two paths $Q_1^a$ and $Q_2^a$ meeting $a$ such that $VC(Q_1^a) \cap \{u_i, v_i\} = \{u_i\}$ and $VC(Q_2^a) \cap \{u_i, v_i\} = \{v_i\}$.

For a root-leaf path $Q$ passing through $a$ denote by $Q_a$ the prefix of $Q$ ending with $a$ and by $\neg Q_a$ the suffix of $Q$ beginning with $a$. Note that by definition of $SV'$, $u_i$ occurs in $P_a$ and $v_i$ occurs in $\neg P_a$. It follows that $u_i$ occurs both in $Q_1^a$ and $Q_2^a$. Indeed, assume w.l.o.g. $u_i$ does not occur in $Q_1^a$. Then, by uniformity of $Z$, $u_i$ occurs in $\neg Q_1^a$. Then $P_a + \neg Q_1^a$ (we denote this way the concatenation of two paths) is a root-leaf path with a double occurrence of $u_i$, a contradiction to $Z$ being read-once. Similarly we establish that $v_i$ occurs in both $\neg Q_1^a$ and $\neg Q_2^a$. It remains to observe that, by definition, $u_i$ occurs
negatively in \( Q_a^2 \) and \( v_i \) occurs negatively in \( \neg Q_a^1 \). Hence \( Q_a^* = Q_a^2 + \neg Q_a^1 \) is a root-leaf path of \( Z \) such that \( VC(Q_a^*) \) is disjoint with \( \{ u_i, v_i \} \), a contradiction to Observation 1 confirming the existence of the desired \( x_i \).

Suppose that there is a root-leaf path \( P' \) of \( Z \) passing through \( a \) such that \( S(a) \notin VC(P') \). This means that there is \( x_i \notin VC(P') \) contradicting the previous two paragraphs. Thus being a \( t \)-node has been established and the lemma follows. ■

For the second statement, let \( A \) and \( B \) be two families of subsets of a universe \( U \). We say that \( A \) covers \( B \) if for each \( S \in B \) there is \( S' \in A \) such that \( S' \subseteq S \). If each element of \( A \) is of size at least \( t \) then we say that \( A \) is a \( t \)-cover of \( B \). Denote by \( VC(G) \) the set of all VCs of \( G \).

**Theorem 4.** There is a function \( f \) such that the following is true. Let \( H \) be a graph. Let \( A \) be a \( t \)-cover of \( VC(H) \). The \( |A| \geq 2^{t/f(x)} \) where \( x \) is the max-degree of \( H \).

The proof of Theorem 4 is provided in Subsection 4.1. Now we are ready to prove Theorem 2.

**Proof of Theorem 2.** Let \( N \) be the set of all \( t \)-nodes of \( Z \). For each \( a \in N \), specify one \( S(a) \) of size at least \( t \) such that for all paths \( P \) of \( Z \) passing through \( a \), \( S(a) \subseteq VC(P) \). Let \( S = \{ S_1, \ldots, S_q \} \) be the set of all such \( S(a) \). Then we can specify \( \text{distinct } a_1, \ldots, a_q \) such that \( S_i = S(a_i) \) for all \( i \in \{ 1, \ldots, q \} \).

Observe that \( S \) is covers \( VC(G) \). Indeed, let \( V' \in VC(G) \). By Observation 1 there is a root-leaf path \( P \) with \( V' = VC(P) \). By Lemma 1 \( P \) passes through some \( a \in N \) and hence \( S(a) \subseteq VC(P) \). By definition, \( S(a) = S_i \) for \( i \in \{ 1, \ldots, q \} \) and hence \( S_i \subseteq V' \). Thus \( S \) is a \( t \)-cover of \( VC(G) \).

It follows from Theorem 4 that \( q = |S| \geq 2^{t/f(x)} \) where \( x \) is a max-degree of \( G \) and \( f \) is a universal function independent on \( G \) or \( t \). It follows that \( Z \) contains at least \( 2^{t/f(x)} \) distinct nodes namely \( a_1, \ldots, a_q \). ■

### 4.1 Proof of Theorem 4

We are going to define a probability distribution of \( VC(G) \) and to show that for a graph \( G \) of constant degree the probability of an element of \( VC(G) \) to be a superset of a specific subset of size at least \( t \) is exponentially small in \( t \). We then conclude that the number of such subsets covering all the elements of \( VC(G) \) must be exponentially large in \( t \). In the technical details that follow, we do not use the probabilities explicitly but rather present the proof in terms of weighted counting.

Let us define a graph \( G \) with fixed vertices as \( (V, E, F) \) where \( V \) and \( E \) bear their usual meaning and \( F \subseteq V \) is the set of fixed vertices. We can also use \( V(G), E(G), F(G) \) to denote \( V, E, F \), respectively. A set \( S \subseteq V(G) \) is a vC of \( G \) if \( S \) is a vC of \( (V, E) \) and in addition, \( F \subseteq S \). Then \( VC(G) \) is the set of all vCs of \( (V, E) \) that contain \( F \) as a subset. We define \( G \setminus v \) as \( (V', E', F') \) with \( (V', E') = (V, E) \setminus v \) (the usual operation of vertex removal from a graph) and \( F' = F \setminus \{ v \} \). We define \( G/v \) as \( (V', E', F'') \), where \( (V', E') \) are as above and \( F'' = F \cup N_G(v) \), where \( N_G(v) \) is the set of neighbours of \( v \) in \( (V, E) \).

Let \( SV \) be a permutation of \( V \). Now we are going to define a decision tree of \( VC(G) \) w.r.t. \( SV \), denoting it by \( T = T_{G,SV} \). It is a rooted binary tree with edges
directed from the parent to a child. If a node \( a \) of \( T \) has two children, we distinguish the left child \( \text{lch}_T(a) \) and the right child \( \text{rch}_T(a) \) (the subscript can be omitted if clear from the context). If \( a \) is a unary node, its only child is considered the left one and the right child is not defined. We denote by \( T_a \) the subtree of \( T \) rooted by \( a \). With this notation in mind we define \( T \) recursively as follows.

If \( G \) is an empty graph then \( T_{G,SV} \) consists of a single node. Otherwise, let \( v_f \) be the first vertex of \( SV \), \( SV' = SV \setminus v_f \) (the suffix of \( SV' \) resulting from the removal of \( v_f \)), and \( rt \) be the root of \( T_{G,SV} \). If \( v_f \in F(G) \) then \( rt \) is a unary node, otherwise \( rt \) is a binary node. The edge \((rt, \text{lch}(rt))\) is labelled with \( v_f \) and \( T_{\text{lch}(rt)} \) is \( T_{G \setminus v_f,SV'} \). If \( rt \) is a binary node (the right child of \( rt \) is defined) then \((rt, \text{lch}(rt))\) is labelled with \( \neg v_f \) and \( T_{\text{rch}(rt)} = T_{G \setminus v_f,SV'} \).

An example of a decision tree as defined above is provided in Figure 4.

![Figure 4](image_url)

**Fig. 4.** A tree \( T_{C,SV C} \) where \( C \) is the graph on the left with \( F(C) = \{v_1\} \) and \( SV C = (v_1, v_2, v_3, v_4) \). All the edges of \( T_{S,SV C} \) are directed to the bottom, hence the arrows on the edges are not shown.

For a root-leaf path \( P \) of \( T \), denote by \( VC(P) \) the set of vertices occurring positively as labels of the edges of \( P \) and let \( P_T \) be the set of all root-leaf paths of \( T \).

**Observation 2** The set \( \{VC(P)|P \in P_T\} \) is precisely \( VC(G) \).

Let \( S \subseteq V \). Denote by \( P_{T,S} \) the set of all root-leaf paths \( P \) of \( T \) such that \( S \subseteq VC(P) \). Let \((a, b)\) be an edge of \( T \) and let \( P \) be a set of paths of \( T \), all starting from \( b \). Then \((a, b) + P = \{(a, b) + P|P \in P\} \) \( (a, b) + P \) denotes the concatenation of a single edge path \((a, b) \) and \( P \).

We say that \( S \) is a distant independent set (DIS) of \( G \) if the distance between any two elements of \( S \) in \( G \) is at least 3 (the vertices of \( S \) are not adjacent and do not have joint neighbours).

**Lemma 2.** Suppose that \( G \) is not empty and let \( v_f \) be the first vertex of \( SV \). Assume that \( S \) is a DIS disjoint with \( F(G) \). Then the following statements are true regarding \( P_{T,S} \).
1. If \( v \in S \) then \( P_{T,S} = (rt, lch(rt)) + P_{T_{lch(rt)}, S \setminus \{vf\}} \).
2. If \( rt \) is a binary node and \( v \) is a neighbour of \( S \) then \( P_{T,S} = (rt, lch(rt)) + P_{T_{lch(rt)}, S \setminus \{vn\}} \) where \( vn \) is the only neighbour of \( v \) in \( S \) (due to \( S \) being a DIS).
3. In all other cases \( P_{T,S} = (rt, lch(rt)) + P_{T_{lch(rt)}, S} \) wherever \( rt \) is a unary node and \( P_{T,S} = [(rt, lch(rt)) + P_{T_{lch(rt)}, S}] \cup [(rt, rch(rt)) + P_{T_{rch(rt)}, S}] \) wherever \( rt \) has two children.

**Proof.** Assume that \( v \in S \) and let \( P \in P_{T,S} \). By our assumption about \( v \), it can occur only as a label on the first edge. Since \( v \in S \), this occurrence must be positive. Consequently, the first edge is \((rt, lch(rt))\). Furthermore, the rest of the labels must be supplied by the suffix of \( P \) starting at \( lch(rt) \). Hence we conclude that this suffix belongs to \( P_{T_{lch(rt)}, S \setminus \{vf\}} \) and hence \( P \in (rt, lch(rt)) + P_{T_{lch(rt)}, S \setminus \{vf\}} \). Conversely, let \( P \in (rt, lch(rt)) + P_{T_{lch(rt)}, S \setminus \{vf\}} \). Then \( v \) occurs positively on the first edge and the rest of vertices of \( S \) occur positively in the subsequent suffix. Thus \( S \subseteq VC(P) \) and hence \( P \in P_{T,S} \).

It is straightforward to observe that if \( v \notin S \) then the third statement holds simply owing to the fact the the occurrences of the vertices of \( S \) are not contributed by the first edges of paths of \( P_T \). However, if \( v \) is a neighbour of \( vn \in S \), it can be noticed that \( P_{T_{vn}, S} = P_{T_{lch(rt)}, S \setminus \{vn\}} \) thus confirming the second statement. Indeed, since \( S \subseteq VC(P) \) implies \( S \setminus \{vn\} \subseteq VC(P) \) for any \( P \in P_T \), \( P_{T_{rch(rt)}, S} \subseteq P_{T_{lch(rt)}, S \setminus \{vn\}} \).

For the opposite direction, recall that \( T_{rch(rt)} = T_{G/vf, SV} \) and \( vn \in F(G/vf) \). This means that \( vn \in VC(P) \) for any path \( P \in P_{T_{rch(rt)}} \). Consequently, \( S \setminus \{vn\} \subseteq VC(P) \) implies that \( S \subseteq VC(P) \) and hence \( P_{T_{rch(rt)}, S \setminus \{vn\}} \subseteq P_{T_{rch(rt)}, S} \).

Let us assign weights to the edges of \( T_{G, SV} \) as follows. For a binary node assign weight 0.5 to both its outgoing edges. For a unary node assign weight 1 to its only outgoing edge. Denote the weight of an edge \( e \) by \( w(e) \). For a path \( P \), the weight \( w(P) \) of \( P \) is a product of weights of its edges, considering the weight of a single vertex path to be 1, and for a set \( P \) of paths, its weight \( w(P) = \sum_{p \in P} w(P) \).

**Observation 3** Let \( a \) be a node of \( T_{G, SV} \). Then the following statements hold.

- \( w(P_a) = 1 \).
- Let \((a, b)\) be an edge of \( T_{G, SV} \) and let \( P \) be a set of paths of \( T_{G, SV} \) all starting from \( b \). Then \( w((a, b) + P) = w((a, b)) * w(P) \).

For \( v \in V(G) \), let \( p_G(v) = 1 - 2^{-d_G(v)+1} \), where \( d_G(v) \) is the degree of \( v \) in \( G \).

The following are simple facts regarding \( p_G(v) \).

**Observation 4** The following statements hold regarding \( p_G(v) \).

- Let \( u \in V(G) \setminus \{v\} \). Then \( p_{G \setminus u}(v) \leq p_G(v) \).
- \( 0.5 \leq p_G(v) \).
- Let \( c \) be the max-degree of \( G \). Then \( p_G(v) \leq 1 - 2^{-(c+1)} \).

The following is the central statement towards the proof of Theorem.\( \square \)
Lemma 3. Let \( S \) be a DIS of \( G \) such that \( S \cap F(G) = \emptyset \), let \( SV \) be an arbitrary permutation of \( V(G) \) and let \( T = T_{G,SV} \). Then \( w(P_{T,S}) \leq \prod_{v \in S} p_G(v) \). (We assume the right-hand part of the inequality to equal 1 if \( S = \emptyset \)).

Proof. By induction on \( |V(G)| \). If \( |S| = 0 \) then the theorem clearly holds because \( w(P_{T,S}) \leq w(P_T) = 1 \) by Observation 3. So, assume that \( |S| > 0 \) and hence \( |V(G)| > 0 \). Let \( rt \) be the root of \( T \) and let \( vf \) be the first vertex of \( SV \).

Suppose \( rt \) is a unary node (this means that \( vf \in F(G) \) and hence \( vf \notin S \)). It follows from Lemma 2 and Observation 3 that \( w(P_{T,S}) = w(P_{T_{lch(rt)},S}) \). Recall that \( T_{lch(rt)} = T_{G \setminus vf,SV \setminus vf} \) and that \( S \) is disjoint with \( F(G \setminus vf) \). Hence, the induction assumption stands. Combining it with the first item of Observation 4, we get

\[
\prod_{v \in S} p_G(v) \leq \prod_{v \in S} p_G(v)
\]

In the rest of the proof we assume that \( rt \) is a binary node. Assume first that \( vf \notin S \cup N(S) \). Then \( S \) remains non-fixed in both \( G \setminus vf \) and \( G/vf \) and hence the induction assumption stands for both \( w(P_{T_{lch(rt)},S}) \) and \( w(P_{T_{rch(rt)},S}) \). Applying the same line of argumentation as in the previous paragraph, we observe that \( w(P_{T_{lch(rt)},S}) \leq \prod_{v \in S} p_G(v) \) and \( w(P_{T_{rch(rt)},S}) \leq \prod_{v \in S} p_G(v) \). By Lemma 2 together with Observation 3 we obtain \( w(P_{T,S}) \leq 0.5 \cdot w(P_{T_{lch(rt)},S}) + 0.5 \cdot w(P_{T_{rch(rt)},S}) \). Substituting \( w(P_{T_{lch(rt)},S}) \) and \( w(P_{T_{rch(rt)},S}) \) with \( \prod_{v \in S} p_G(v) \), we obtain \( w(P_{T,S}) \leq 0.5 \cdot \prod_{v \in S} p_G(v) + 0.5 \cdot \prod_{v \notin S} p_G(v) \) as required.

Assume now that \( vf \in S \). Observe that \( S \setminus \{vf\} \) is not fixed in \( G \setminus vf \). Hence, arguing as is the previous two paragraphs, we conclude that \( w(P_{T_{lch(rt)},S \setminus \{vf\}}) \leq \prod_{v \in S \setminus \{vf\}} p_G(v) \). Lemma 2 together with Observation 3 yield \( w(P_{T,S}) \leq 0.5 \cdot w(P_{T_{lch(rt)},S \setminus \{vf\}}) \). Substituting \( w(P_{T_{lch(rt)},S \setminus \{vf\}}) \), we obtain \( w(P_{T,S}) \leq 0.5 \cdot \prod_{v \in S \setminus \{vf\}} p_G(v) \). By the second item of Observation 4, 0.5 can be replaced by \( p_G(vf) \) in the last inequality. That is \( w(P_{T,S}) \leq p_G(vf) \prod_{v \in S \setminus \{vf\}} p_G(v) = \prod_{v \in S} p_G(v) \) as required.

Finally, suppose that \( vf \) is a neighbour of \( S \). That is \( vf \) is a neighbour of exactly one vertex \( vn \in S \). Observe that \( S \) is not fixed in \( G \setminus vf \) and \( S \setminus \{vn\} \) is not fixed in \( G/vf \). Hence, arguing as above, we conclude that \( w(P_{T_{lch(rt)},S \setminus \{vn\}}) \leq p_G(vf)(vn) \prod_{v \in S \setminus \{vn\}} p_G(v) \) and that \( w(P_{T_{rch(rt)},S \setminus \{vn\}}) \leq \prod_{v \in S \setminus \{fn\}} p_G(v) \) (notice that we have not replaced \( p_G(vf)(vn) \) by \( p_G(vn) \) as retaining the former is essential for the forthcoming reasoning). By Lemma 2 and Observation 3 we have \( w(P_{T,S}) \leq 0.5 \cdot w(P_{T_{lch(rt)},S \setminus \{vn\}}) + 0.5 \cdot w(P_{T_{rch(rt)},S \setminus \{vn\}}) \). Substituting \( w(P_{T_{lch(rt)},S \setminus \{vn\}}) \) and \( w(P_{T_{rch(rt)},S \setminus \{vn\}}) \) and moving \( \prod_{v \in S \setminus \{vn\}} p_G(v) \) outside the brackets, we obtain \( w(P_{T,S}) \leq 0.5 \cdot p_G(vf)(vn) + 1) \prod_{v \in S \setminus \{vn\}} p_G(v) \). The last step of our reasoning is the observation that \( 0.5 \cdot (p_G(vf)(vn) + 1) = p_G(vn) \). Indeed, \( p_G(vn) = (1 - 2^{-d_G(vn) + 1}) = 0.5(2 - 2^{-d_G(vn)} + 1) = 0.5(2 - 2^{-d_G(vf)(vn) + 1}) = 0.5(p_G(vf)(vn) + 1) \). Thus \( w(P_{T,S}) \leq p_G(vn) \prod_{v \in S \setminus \{vn\}} p_G(v) = \prod_{v \in S} p_G(v) \) as required.

Proof of Theorem 4. To consider \( H \) in the theorem statement as a graph with fixed vertices, we represent it as \( (V,E,\emptyset) \). Let \( SV \) be an arbitrary permutation of \( V(H) \) and let \( T = T_{H,SV} \).

For the given integer \( x > 0 \), let \( ax \) be the constant such that \( 2^{-1/ax} = (1 - 2^{-x+1}) \). Let \( c \) be the max-degree of \( H \). Then, by the last statement of Observation 4, for any \( v \in V(H) \), \( p_H(v) \leq 2^{-1/ax} \).
Let $S$ be a DIS of $H$. Then, combining the previous paragraph with Lemma 3, we observe that $w(P_{T,S}) \leq 2^{-|S|/a_c}$.

Let $S^*$ be an arbitrary subset of $V(H)$. Observe that there is a DIS $S \subseteq S^*$ of size at least $|S^*|/(c^2 + 1)$. Indeed, let $S \subseteq S^*$ be a largest DIS among the subsets of $S$. Then each element of $S^* \setminus S$ is at distance at most 2 from an element of $S$. For each $u \in S$, there are at most $c + c(c-1) = c^2$ elements of $H$ lying at distance at most 2 from $S$. Thus $|S^* \setminus S| \leq |S| \cdot c^2$, that is $|S^*| \leq |S| \cdot (c^2 + 1)$ and hence $|S| \geq |S^*|/(c^2 + 1)$. Since $P_{T,S^*} \subseteq P_{T,S}$, $w(P_{T,S^*}) \leq w(P_{T,S}) \leq 2^{-|S|/b_c}$, where $b_c = a_c \cdot (c^2 + 1)$.

Let $S_1, \ldots, S_q$ be a $t$-cover of $VC(H)$. This means that for each $P \in P_T$ there is $S_i$ whose vertices occur as positive labels on $P$. In other words, $P_T = \bigcup_{i=1}^q P_{T,S_i}$. Hence $1 = w(P_T) \leq \sum_{i=1}^q w(P_{T,S_i}) \leq q \cdot 2^{-t/b_c}$, where the first equality follows from Observation 3. Consequently, $q \geq 2^{t/b_c}$ as claimed. ■

5 Proof of Theorem 3

Denote by $T_r$ a complete binary tree of height (root-leaf distance) $r$. Let $T$ be a tree and $H$ be an arbitrary graph. Then $T(H)$ is a graph having disjoint copies of $H$ in one-to-one correspondence with the vertices of $T$. For each pair $t_1, t_2$ of adjacent vertices of $T$, the corresponding copies are connected by making adjacent the pairs of same vertices of these copies. Put differently, we can consider $H$ as a labelled graph where all vertices are associated with distinct labels. Then for each edge $\{t_1, t_2\}$ of $T$, edges are introduced between the vertices of the corresponding copies having the same label. An example of this construction is shown on Figure 5.

![Figure 5](image.png)

Fig. 5. Graphs from the left to the right: $T_3$, $P_3$, $T_3(P_3)$. The dotted ovals surround the copies of $P_3$ in $T_3(P_3)$.

In order to prove Theorem 3, we first prove three structural lemmas about graphs $T(H)$, the first one being an auxiliary statement for the second one and the second one being an auxiliary statement for the third one.

**Lemma 4.** Suppose the vertices of $T(H)$ are partitioned into 2 subsets $V_1$ and $V_2$. Let $L$ be a subset of vertices of $H$ such that $|L| = t$. Suppose there are two copies $H_1$ and $H_2$ of $H$ such that for each $u \in L$ the copies of vertex $u$ in $H_1$ and $H_2$ belong to different partition classes. Then $T(H)$ has matching of size $t$ with the ends of each edge lying in different partition classes.
Lemma 5. Let $T$ be a tree consisting of at least $p$ vertices. Let $H$ be a connected graph of at least $2p$ vertices. Let $V_1, V_2$ be a partition of $V(T(H))$ such that both partition classes contain at least $p^2$ vertices. Then $T(H)$ has a matching of size $p$ with the ends of each edge belong to distinct partition classes.

Proof. Assume first that there are at least $p$ copies of $H$ corresponding to vertices of $T$ that contain vertices of both partition classes. Since $H$ is a connected graph, for each copy we can specify an edge with one end in $V_1$ and the other end in $V_2$. These edges belong to disjoint copies of $H$, hence none of these edges have a common end. Since there are $p$ copies of $H$, we have the desired matching of size $p$.

If the assumption in the previous paragraph is not true then, since $T$ has at least $p$ vertices, there is a vertex $u$ of $T$ such that the copy $H_1$ of $H$ corresponding to $u$ contains vertices of only one partition class; assume w.l.o.g. that this class is $V_1$. We call $u$ a non-partitioned vertex of $T$. Then there is a vertex $v$ of $T$ such that the copy $H_2$ of $H$ corresponding to $v$ contains at least $p$ vertices of $V_2$. Indeed, otherwise, the vertices of the copies of $H$ associated with the non-partitioned vertices of $T$ all belong to $V_1$. Consequently, vertices of $V_2$ can occur only in the remaining at most $p-1$ copies of $H$. If each of these copies contains at most $p-1$ vertices of $V_2$ then the total number of vertices of $V_2$ is smaller than $p^2$ in contradiction to our assumption. We conclude that the required vertex $v$ indeed exists.

Let $L$ be the set of vertices of $H$ whose copies in $H_2$ belong to $V_2$. By assumption, all the copies of $L$ in $H_1$ belong to $V_1$. By Lemma 4, $H_1$ and $H_2$ witness the existence of matching of size $p$ with ends of each edge belonging to distinct partition classes. ■

Lemma 6. Let $p$ be an arbitrary integer and let $H$ be an arbitrary connected graph of $2p$ vertices. Then for any $r \geq \lceil \log p \rceil$, $mw(T_r(H)) \geq (r + 1 - \lceil \log p \rceil)p/2$.

Proof. The proof is by induction on $r$. The first considered value of $r$ is $\lceil \log p \rceil$. After that $r$ will increment in 2. In particular, for all values of $r$ of the form $\lfloor \log p \rfloor + 2x$, we will prove that $mw(T_r(H)) \geq (x + 1)p$ and, moreover, for each permutation $SV$ of $V(T_r(H))$, the required matching can be witnessed by a partition of $SV$ into a suffix and a prefix of size at least $p^2$ each. Let us verify that the lower bound $mw(T_r(H)) \geq (x + 1)p$ implies the lemma. Suppose that $r = \lceil \log p \rceil + 2x$ for some non-negative integer $x$. Then $mw(G) \geq (x + 1)p = ((r - \lceil \log p \rceil)/2 + 1)p > (r - \lceil \log p \rceil + 1)p/2$. Suppose $r = \lceil \log p \rceil + 2x + 1$. Then $mw(G) = mw(T_r(H)) \geq mw(T_{r-1}(H)) \geq (x + 1)p = ((r - \lceil \log p \rceil - 1)/2 + 1)p = (r - \lceil \log p \rceil + 1)p/2$. 

Proof. Let $v_1$ and $v_2$ be the respective vertices of $T$ corresponding to $H_1$ and $H_2$. Let $p$ be the path between $v_1$ and $v_2$ in $T$. Then for each $u \in L$ there are two consecutive vertices $v_1'$ and $v_2'$ of this path with respective copies $H_1'$ and $H_2'$ such that the copy $u_1'$ of $u$ in $H_1'$ belongs to the same partition class as the copy $u_1$ of $u$ in $H_1$ and the copy $u_2'$ of $u$ in $H_2'$ belongs to the same partition class as the copy $u_2$ of $u$ in $H_2$. By construction, $T(H)$ has an edge $\{u_1', u_2'\}$ which we choose to correspond to $u$. Let $L = \{u^1, \ldots, u^t\}$ and consider the set of edges as above corresponding to each $u^t$. By construction, both ends of the edge corresponding to each $u^t$ are copies of $u^t$ and also these ends correspond to distinct partition classes. It follows that these edges do not have joint ends and indeed constitute a desired matching of size $t$. ■
Assume that \( r = \lceil \log p \rceil \) and let us show the lower bound of \( p \) on the matching width. \( T_r \) contains at least \( 2^{\lceil \log p \rceil + 1} - 1 \geq 2^{\log p + 1} - 1 = 2p - 1 \geq p \) vertices. By construction, \( H \) contains at least \( 2p \) vertices. Consequently, for each ordering of vertices of \( T_r \) we can specify a prefix and a suffix of size at least \( p^2 \) (just choose a prefix of size \( p^2 \)). Let \( V_1 \) be the set of vertices that got to the prefix and let \( V_2 \) be the set of vertices that got to the suffix. By Lemma 5, there is a matching of size at least \( p \) consisting of edges between \( V_1 \) and \( V_2 \) confirming the lemma for the considered case.

Let us now prove the lemma for \( r = \lceil \log p \rceil + 2x \) for \( x \geq 1 \). Specify the centre of \( T_r \) as the root and let \( T^1, \ldots, T^4 \) be the subtrees of \( T_r \) rooted by the grand-children of the root. Clearly, all of \( T^1, \ldots, T^4 \) are copies of \( T_{r-2} \). Let \( SV \) be a sequence of vertices of \( V(T_r(H)) \). Let \( SV^1, \ldots, SV^4 \) be the respective sequences of \( V(T^1(H)), \ldots, V(T^4(H)) \) ‘induced’ by \( SV \) (that is their order is as in \( SV \)). By the induction assumption, for each of them we can specify a partition \( SV^1, SV^2 \) into a prefix and a suffix of size at least \( p^2 \) each witnessing the conditions of the lemma for \( r - 2 \).

Let \( u_1, \ldots, u_4 \) be the last respective vertices of \( SV^1, \ldots, SV^4 \). Assume w.l.o.g. that these vertices occur in \( SV \) in the order they are listed. Let \( SV', SV'' \) be a partition of \( SV \) into a prefix and a suffix such that the last vertex of \( SV' \) is \( u_2 \). By the induction assumption we know that the edges between \( SV^2 \subseteq SV' \) and \( SV^2 \subseteq SV'' \) form a matching \( M \) of size at least \( xp \). In the rest of the proof, we are going to show that the edges between \( SV' \) and \( SV'' \) whose ends do not belong to any of \( SV^2 \subseteq SV' \), \( SV^2 \subseteq SV'' \) can be used to form a matching \( M' \) of size \( p \). The edges of \( M \) and \( M' \) do not have joint ends, hence this will imply existence of a matching of size \( xp + p = (x + 1)p \), as required.

The sets \( SV' \setminus SV^2 \) and \( SV'' \setminus SV^2 \) partition \( V(T_r(H)) \setminus (SV^2 \cup SV^2) = V(T_r(H)) \setminus V(T^2(H)) = V(T_r \setminus T^2)(H) \). Clearly, \( T_r \setminus T^2 \) is a tree. Furthermore, it contains at least \( p \) vertices. Indeed, \( T^2 \) (isomorphic to \( T_{r-2} \)) has \( p \) vertices just because we are at the induction step and \( T_r \) contains at least \( 4 \times \) more vertices than \( T^2 \). So, in fact, \( T_r \setminus T^2 \) contains at least \( 3p \) vertices. Furthermore, since \( u_1 \) precedes \( u_2 \), the whole \( SV^2 \) is in \( SV' \). By definition, \( SV^2 \) is disjoint with \( SV^2 \) and hence it is a subset of \( SV' \setminus SV^2 \). Furthermore, by definition, \( |SV^2| \geq p^2 \) and hence \( |SV' \setminus SV^2| \geq p^2 \) as well. Symmetrically, since \( u_2 \in SV'' \), we conclude that \( SV^2 \subseteq SV'' \setminus SV^2 \) and due to this \( |SV'' \setminus SV^2| \geq p^2 \).

Thus \( SV' \setminus SV^2 \) and \( SV'' \setminus SV^2 \) partition \( V(T_r \setminus T^2)(H) \) into classes of size at least \( p^2 \) each and the size of \( T_r \setminus T^2 \) is at least \( 3p \). Thus, according to Lemma 5, there is a matching \( M' \) of size at least \( p \) created by edges between \( SV' \setminus SV^2 \) and \( SV'' \setminus SV^2 \), confirming the lemma, as specified above.

**Proof of Theorem 3.** First of all, let us identify the class \( G \). Recall that \( P_x \) a path of \( x \) vertices. Further on, let \( 0 \leq y \leq 3 \) be such that \( k - y + 1 \) is divided by \( 4 \). The considered class \( G \) consists of all \( G = T_r(P_{x-y+1}) \) for \( x \geq 5 \lceil \log k \rceil \).

Let us show that the treewidth of the graphs of \( G \) is bounded by \( k \). Consider the following tree decomposition of \( G = T_r(H = P_{x-y+1}) \). The tree is \( T_r \). Consider \( T_r \) as the rooted tree with the centre being the root. The bag of each vertex includes the vertices of the copy of \( H \) associated with this vertex plus the copy of the parent (for a non-root vertex). The properties of tree decomposition can be verified by a direct inspection. The size of each bag is at most \( k - y \), hence the treewidth is at most \( k - y \leq k \).
Observe that max-degree of the graphs of G is 5. Indeed, consider a vertex v of G ∈ G that belongs to a copy of H associated with a vertex x of some T_r. Inside its copy of H, v is adjacent to at most 2 vertices. Outside its copy of H, v is adjacent to vertices in the copies of H associated with the neighbours of x, precisely one neighbour per copy. Vertex x is adjacent to at most 3 vertices of T_r. It follows that v has at most 3 neighbours outside its copy of H.

Let us reformulate the lower bound of mw(G) in terms of logn and k where n = V(G). Notice that p used in Lemma 5 can be expressed as (k − y + 1)/4. Hence, the lower bound on the matching width can be seen as (r − ⌈log((k−y+1)/4)⌉ + 1) * (k − y + 1)/8. This lower bound can be immediately simplified by noticing that by the choice of k and y, (k − y + 1)/8 ≥ k/16 and ⌈log((k−y+1)/4)⌉ ≤ ⌈logk⌉. Hence, (r − ⌈logk⌉ + 1)k/16 can serve as a lower bound on mw(G). To draw the connection between n and r, notice that n = (2^{r+1} − 1)(k − y + 1)/2. It follows that r + 1 = log((n n/2)/k − 1) + 1)(k − y + 1)/4 + 1). In particular, it follows that r + 1 ≥ logn − logk ≥ logn − ⌈logk⌉. It follows that r + 1 in the lower bound can be replaced by logn − ⌈logk⌉ and the new lower bound is (logn − 2[logk])k/16. Consequently, for logn ≥ 5[logk] the lower bound can be represented as (logn * k)/32 which is the form needed for the theorem. It remains to observe that r ≥ 5[logk] implies logn ≥ 5[logk]. By the above reasoning, r ≥ 5[logk] implies log((n n/2)/2 + 1) ≥ 5[logk]. By our choice of k ≥ 50, log(n/20 + 1) ≥ log((n n/2)/2 + 1) ≥ 5[logk]. By construction of G and the choice of r, n ≥ 2^{r+1} − 1 ≥ k^5 − 1 ≥ k, the last inequality follows from the choice of k, hence n ≥ 50. In particular, it follow that n ≥ n/20 + 1. Hence logn ≥ log(n/20 + 1) ≥ 5[logk].

6 Separation between ROBP and decision DNNF

Lemma 7. The space complexity of aroSRN on CNFs φ(T_r(P_r)) is Ω(n log n/c) for some universal constant c.

Proof. The number of variables of T_r(P_{2r}) is n = (2^{r+1} − 1) * 2r = 2^{r+1} * 4r − 2r. That is, r = log((2^{r+1} − 1) * 2r) = log n − log r − 2. For a sufficiently large r, r ≥ log r + 2, hence r ≥ log n − r and hence r ≥ log n/2.

By Lemma 6, mw(T_r(P_{2r})) ≥ (r + 1 − ⌈logr⌉)r/2. That is, mw(T_r(P_{2r})) ≥ (log n − 1 − log log n − 1) * logn/4. It is not hard to see that for a sufficiently large r (and hence sufficiently large n), mw(T_r(P_{2r})) ≥ log^2 n/16. The statement of the theorem now follows immediately from Theorem 2.

Theorem 5. There is an infinite class of CNFs such that the complexity of decision DNNF on this class is O(n^5) while the complexity ROBP is Ω(n log n/c) for some universal constant c.

Proof. Consider the class φ(T_r(P_r)). As a ROBP can be seen as a special case of an aroSRN, the lower bound on the space complexity of ROBP on φ(T_r(P_r)) immediately follows from Lemma 7.

It follows from Theorem 1 in [10] that the space complexity of decision DNNF on a CNF of primal graph treewidth t is O(2^t n) (the theorem in fact uses a different
parameter of a CNF, however it is shown to never exceed the primal graph treewidth). Arguing as in the proof of Theorem 3, we observe that the treewidth of \( T_r(P_{2r}) \) is at most \( 4r \). We know from the proof of Lemma 7 that \( r = \log \frac{n + 2r}{4r} \). That is \( r \leq \log(n + 2r) \) and, for a sufficiently large \( r \), \( r \leq \log(2n) = \log n + 1 \). That is, for a sufficiently large \( r \), the treewidth of \( T_r(P_{2r}) \) is \( 4\log n + 4 \). Substituting \( 4\log n + 4 \) instead \( t \) in \( O(2^n) \) results in \( O(n^5) \), completing the required separation. ■

Thus, Theorem 5 shows that the quasi-polynomial upper bound on the size of ROBP simulating the given decision DNNF as described in 1 is essentially tight.

References

1. Paul Beame, Jerry Li, Sudeepa Roy, and Dan Suciu. Lower bounds for exact model counting and applications in probabilistic databases. In Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, Bellevue, WA, USA, August 11-15, 2013, 2013.
2. Randal E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. ACM Comput. Surv., 24(3):293–318, 1992.
3. Adnan Darwiche. Decomposable negation normal form. J. ACM, 48(4):608–647, 2001.
4. Adnan Darwiche. SDD: A new canonical representation of propositional knowledge bases. In IJCAI, pages 819–826, 2011.
5. Adnan Darwiche and Pierre Marquis. A knowledge compilation map. J. Artif. Intell. Res. (JAIR), 17:229–264, 2002.
6. Andrea Ferrara, Guoqiang Pan, and Moshe Y. Vardi. Treewidth in verification: Local vs. global. In LPAR, pages 489–503, 2005.
7. Abhay Kumar Jha and Dan Suciu. On the tractability of query compilation and bounded treewidth. In ICDT, pages 249–261, 2012.
8. Stasys Jukna. Boolean Function Complexity: Advances and Frontiers. Springer-Verlag, 2012.
9. Kenneth L. McMillan. Hierarchical representations of discrete functions, with application to model checking. In CAV, pages 41–54, 1994.
10. Umut Oztok and Adnan Darwiche. Cw-width: A new complexity parameter for cnfs. In ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014), pages 675–680, 2014.
11. Alexander A. Razborov. Lower bounds for deterministic and nondeterministic branching programs. In FCT, pages 47–60, 1991.
12. Alexander A. Razborov, Avi Wigderson, and Andrew Chi-Chih Yao. Read-once branching programs, rectangular proofs of the pigeonhole principle and the transversal calculus. In STOC, pages 739–748, 1997.
13. Igor Razgon. On OBDDs for CNFs of bounded treewidth. In KR, pages 92–100, 2014.
14. Igor Razgon and Justyna Petke. Cliquewidth and knowledge compilation. In SAT, pages 335–350, 2013.
15. Martin Vatshelle. New width parameters of graphs. PhD thesis, Department of Informatics, University of Bergen, 2012.
16. Ingo Wegener. Branching Programs and Binary Decision Diagrams. SIAM, 2000.

A Transformation of an AROSRN into a uniform one

Let \( Z \) be the AROSRN being transformed and let \( F \) be the function of \( n \) variables realized by \( Z \). Let \( a_1, \ldots, a_m \) be the non-leaf nodes of \( Z \) being ordered topologically. We
show that there is a sequence $Z_{a_1} = Z, Z_{a_2}, \ldots, Z_{a_m}$ such that each $Z_{a_i}$ for $i > 1$ is an arosrn of $F$ obtained from $Z_{a_{i-1}}$ by subdividing the in-coming edges of $a_i$ by adding at most $n$ nodes and $O(n)$ edges to each such an in-coming edge. Moreover, the edges of any two paths $P_1$ and $P_2$, starting the root of $Z_{a_i}$ and ending at the same node which is ether $a_i$ or topologically precedes $a_i$, are labelled with literals of the same set of variables. Observe that since each edge has only one head, say $a_j$, it is subdivided only once, namely during the construction of $Z_{a_j}$. Hence the number of new added edges of $Z_{a_m}$ is $O(n)$ per edge of $Z$ and hence the size of $Z_{a_m}$ is $O(n)$ times larger than the size of $Z$.

Regarding $Z_{a_i}$, this existence statement is vacuously true so assume $i > 1$ Denote by $Var(a_i)$ the set of all variables whose literals label edges of paths of $Z_{a_{i-1}}$ from the root to $a_i$.

For each in-neighbour $a'$ of $a_i$, we transform the edge $(a', a_i)$ as follows. Let $P$ be a path from the root of $Z_{a_{i-1}}$ to $a_i$ through $(a', a_i)$. Let $x^1, \ldots, x^q$ be the elements of $Var(a_i) \setminus Val(A(P))$. We subdivide $(a', a_i)$ as follows. We introduce new nodes $a'_1, a'_2, \ldots, a'_m$ and let $a'_{i+1} = a_i$. Then instead $(a', a_i)$ we introduce an edge $(a', a'_i)$ carrying the same label as $(a', a_i)$ (or no label in case $(a', a_i)$ carries no label). Then, for each $1 \leq i \leq q$ we introduce two edges $(a'_i, a'_{i+1})$ carrying labels $x^q$ and $\neg x^q$, respectively.

Let us show that the edges of any two paths $P_1$ and $P_2$ from the root of $Z_{a_i}$ to $a_i$ are labeled with literals of the same set of variables. Let $a'$ be an in-neighbour of $a_i$ in $Z_{a_{i-1}}$. By the induction assumption, any two paths from the root to $a'$ are labelled with literals of the same set of variables. It follows that as a result any two paths from the root to $a'_i$ passing through $a'$ are labelled by literals of the same set of variables, namely $Var(a_i)$. Since this is correct for an arbitrary choice of $a'$, we conclude that in $Z_{a_i}$ any two paths from the root to $a_i$ are labelled with $Var(a_i)$, that is with literals of the same set of variables. Observe that the paths to the nodes of $Z$ preceding $a_i$ are not affected so the `uniformity' of paths regarding them holds by the induction assumption. Regarding the new added nodes on the subdivided edge $(a', a_i)$ the uniformity clearly follows from the uniformity of paths from the root to $a'$.

To verify the read-once property of $Z_{a_i}$, let $P'$ be a path from the root to the leaf of $Z_{a_i}$. Taking into account the induction assumption, the only reason why $P'$ may contain two edges labelled by literals of the same variable is that $P'$ is obtained from a path $P$ of $Z_{a_{i-1}}$ by subdivision of an edge $(a', a_i)$ of this path. By construction the variables of the new labels put on $(a', a_i)$ do not occur on the prefix of $P$ ending at $a_i$. Furthermore, by definition of $Var(a_i)$ the variable $x$ of each new label, in fact, occurs in some path of $Z_{a_{i-1}}$ from the root to $a_i$ and hence, by the read-once property, $x$ does not occur on any path starting from $a_i$. It follows that the variables of the new labels do not occur on the suffix of $P'$ starting at $a_i$. Taking into account that all the new labels of $(a', a_i)$ are literals of distinct variables, the read-once property of $P'$, and hence the read-once property of $Z_{a_i}$, due to the arbitrary choice of $P'$, follow. Thus we know now that $Z_{a_i}$ is an arosrn.

It remains to verify that $Z_{a_i}$ indeed realizes $F$. Let $P'$ be a path of $Z_{a_i}$ from the root to the leaf. Then $A(P')$ is an extension of $A(P)$ of some path $P$ of $Z_{a_i}$. By the induction assumption, any extension of $A(P)$ is a satisfying assignment of $F$, hence so
is \( A(P') \). Conversely, for each satisfying assignment \( A \) of \( F \) we can find a path \( P \) of \( Z_{n-1} \) such that \( A(P) \subseteq A \). If an edge of path \( P \) is subdivided then the new labels are opposite literals on multiple edges. So, for every such multiple edge we can choose one edge carrying the literal occurring in \( A \) and obtain a path \( P' \) such that \( A(P') \subseteq A \).

For the leaf node we do a similar transformation but this time add new labels on the in-coming edges of the leaf so that the set of labels on each path from the root to the leaf is a set of literals of \( Var(F) \). A similar argumentation to the above shows that the resulting structure is indeed a uniform AROSRN realizing \( F \). Clearly the size of the resulting AROSRN remains \( O(n) \) times larger than the size of \( Z \).