The splitting power of branching programs of bounded repetition and CNFs of bounded width

Igor Razgon [*]
Department of Computer Science and Information Systems,
Birkbeck University of London
i.razgon@bbk.ac.uk

Abstract

In this paper we study syntactic branching programs of bounded repetition representing CNFs of bounded treewidth. For this purpose we introduce two new structural graph parameters d-pathwidth and clique preserving d-pathwidth denoted by $pw_d(G)$ and $cpw_d(G)$ where G is a graph. We show that $cpw_2(G) \leq O(tw(G)\Delta(G))$ where $tw(G)$ and $\Delta(G)$ are, respectively the treewidth and maximal degree of G. Using this upper bound, we demonstrate that each CNF ψ can be represented as a conjunction of two OBDDs (quite a restricted class of read-twice branching programs) of size $2^{O(\Delta(\psi) \cdot tw(\psi)^2)}$ where $tw(\psi)$ is the treewidth of the primal graph of ψ and each variable occurs in ψ at most $\Delta(\psi)$ times.

Next, we use d-pathwidth to obtain lower bounds for monotone branching programs. In particular, we consider the monotone version of syntactic nondeterministic read d times branching programs (just forbidding negative literals as edge labels) and introduce a further restriction that each computational path can be partitioned into at most d read-once subpaths. We call the resulting model separable monotone read d times branching programs and abbreviate them d-SMNBPs. For each graph G without isolated vertices, we introduce a CNF $\psi(G)$ whose clauses are $(u \vee e \vee v)$ for each edge $e = \{u, v\}$ of G. We prove that a d-SMNBP representing $\psi(G)$ is of size at least $\Omega(c^{pw_d(G)})$ where $c = (8/7)^{1/12}$. We use this 'generic' lower bound to obtain an exponential lower bound for a 'concrete' class of CNFs $\psi(K_n)$. In particular, we demonstrate that for each $0 < a < 1$, the size of n^a-SMNBP representing $\psi(K_n)$ is at least c^{n^b} where b is an arbitrary constant such that $a+b < 1$. This lower bound is tight in the sense $\psi(K_n)$ can be represented by a poly-sized n-SMNBP.

Keywords: branching programs, tree-partition width, treewidth.

[*]Declaration of interests: none.

1 Introduction

1.1 The starting point

The main aim of the present paper is to contribute to the resolution of the following open question.

Open Question 1 *Let $d > 1$ be a constant. Is there a function f and a constant a (possibly both dependent on d) so that a CNF of primal treewidth at most k can be represented as a read d-times nondeterministic branching program (d-NBP) of size at most $f(k) \cdot n^a$?*

The function $F(n,k) = f(k) \cdot n^a$ is *FPT*: it is $O(n^a)$ for every *fixed* k. This is opposed to an *XP* function $G(n,k) = n^{g(k)}$ which is also polynomial for every fixed k but the degree of the polynomial may grow with the growth of k. It is known that a CNF of bounded treewidth can be represented by an XP sized 1-NBP and that the upper bound is essentially tight [14]. In other words, the XP lower bound proved in [14] rules out an FPT upper bound for representation of CNFs of bounded treewidth by d-NBPs if $d = 1$. It is thus natural to ask if an FPT upper bound is possible if we increase d. Open Question 1 asks exactly that.

In fact, we believe that the resolution of this question is of a broader interest as it leads to challenging open questions in the areas of graph theory and circuit complexity. Later on in this paper, we will provide evidence supporting this point of view.

We positively answer Open Question 1 under the bounded degree assumption (each variable of the input CNF appears in a bounded number of clauses) establishing an FPT upper bound for a restricted version of 2-NBP.

For the unbounded degree we consider *monotone* d-NBP obeying one additional restriction that we call *separability*. On the one hand, these conditions are quite general (the monotonicity is, arguably, one of the most common assumptions in the area of circuit complexity, for the separability, see Remark 2 in Section 2). On the other hand, we demonstrate that under this restriction, the resolution of Open Question 1 reduces to resolution of a purely graph theoretical question of dependency on treewidth of a novel parameter introduced in this paper. Our final result a lower bound for the monotone separable nondeterministic branching programs that, in our opinion, demonstrates that this model is interesting in a broader circuit complexity context.

In the rest of this section we provide detailed description of the proposed results (Section 1.2), support these results with detailed motivation (Section 1.3), and outline the structure of the rest of the paper (Section 1.4).

1.2 Statement of results

In this paper we study representation of CNFs with bounded structural width parameters by syntactic branching programs of bounded repetition. It is known that Nondeterministic read-once branching programs (1-NBPs) in general require at least $n^{\Omega(k)}$ size to represent CNFs of primal treewidth k even if each variable occurs at most 5 times [14]. However, already read-twice branching programs have a *splitting power* allowing them to efficiently represent a CNF $\varphi = \varphi_1 \wedge \varphi_2$ provided that each CNF φ_i can be efficiently represented by a read-once branching program: simply identify the source of one read-once branching program with the 'positive' sink of the other.

It turns out that the splitting power is sufficient for efficient representation of CNFs of bounded treewidth provided that each variable occurs a bounded number of times. In particular, in Section 4 we prove that a CNF φ of primal treewidth at most k and in which each variable occurs at most d times can be represented by a conjunction of Ordered Binary Decision Diagrams (OBDDs) of size at most $2^{O(dk^2)}n$ each.

The above result is based on a graph-theoretical argument. In particular, we introduce two new parameters of a graph G, d-pathwidth denoted by $pw_d(G)$ and clique-preserving d-pathwidth denoted by $cpw_d(G)$. The d-pathwidth of G is the smallest k such that there are graphs $G_1, \dots, G_d$ of pathwidth at most k such that $G = G_1 \cup \dots \cup G_d$. The clique preserving variant puts an extra requirement that each clique of G is a subgraph of some G_i. We show that $cpw_2(G)$ is linearly upper bounded by the *tree-partition width* of G. The latter parameter is known to be $O(\Delta(G) \cdot tw(G))$ where $\Delta(G)$ is the maximal degree of G and $tw(G)$ is the treewidth of G [19]. It follows that $cpw_2(G) \leq O(\Delta(G) \cdot tw(G))$. Further on, it is known [8] that a CNF φ can be represented by an OBDD of size $2^{pw(G_\varphi)}$ where G_φ is the primal graph of φ and pw denotes the pathwidth. We demonstrate that it follows that φ can be represented as a conjunction of two OBDDs of size $2^{O(cpw_2(G_\varphi))} \leq 2^{O(\Delta(G_\varphi) \cdot tw(G_\varphi))}$. We next observe that each variable appears in at most d clauses of size at most $k+1$ each (because $tw(G_\varphi) \leq k$). We conclude that $\Delta(G_\varphi) \leq d \cdot k$ and hence $2^{O(\Delta(G_\varphi) \cdot tw(G_\varphi))} \leq 2^{dk \cdot k}$ as required.

In Section 5, we consider the following question. Suppose that for a class of CNFs the d-pathwidths of their primal graphs is *at least* k (to put it informally, the splitting power cannot be applied). Does this lower bound imply an exponential in k lower bound for read d times branching programs? We answer this question positively for a quite general subclass of *monotone* nondetermnistic read d times branching programs (d-MNBPs) that we call *separable* and abbreviate d-SMNBP. We describe this result in the following two paragraphs.

The restriction on d-MNBPs imposed by d-SMNBPs is that the sequence of variables along each each source-sink path can be partitioned into at most d read-once fragments. That is to say, if $d = 2$ then querying variables like $x_1, x_2, x_3, x_3, x_2, x_4, x_1$ is allowed while the querying x_1, x_1, x_2, x_2 is not. The order of variables along different parts may be different, in particular a conjunction of d 1-MNBPs is a special case of d-SMNBP. Note that this way of querying generalizes Indexed Binary Decision Diagrams (d-IBDDs) [3] that have an extra requirement of being oblivious.

Let G be a graph without isolated vertices. A CNF $\psi(G)$ corresponding to G has $V(G) \cup E(G)$ as the set of variables. The clauses of $\psi(G)$ are of the form $(u \vee e \vee v)$ for each edge $e = \{u, v\}$ of G. Essentially, each clause $(u \vee v)$ is padded with a unique extra variable. We prove that if $pw_d(G)$ is at least k then the size of a d-SMNBP representing $\psi(G)$ is at least $\Omega(c^k)$ where $c = (8/7)^{1/12}$. Note that the lower bound is *scalable* in the sense that it does not depend on d.

We apply the above statement to obtain a lower bound for a concrete class of CNFs: $\psi(K_n)$. It is easy to see that $\psi(K_n)$ can be represented by a polynomial size n-SMNBP (in fact, by a conjunction of n 1-NBPs). However, reducing the allowed number of repetitions from n to n^a for an arbitrary constant $0 < a < 1$ results in an exponential lower bound. In particular, combining the $\Omega(c^k)$ lower bound with an upper bound on the number of edges in a graph of bounded pathwidth, we demonstrate that an n^a-SMNBP representing $\psi(K_n)$ has size at least $\Omega(c^{(n^b)})$. where b is an arbitrary constant such that $a + b < 1$.

The above lower bound gives rise to several lines of further research. First, we conjecture that w.r.t. the treewidth alone (without bounded maximal degree assumption), d-pathwidth behaves just like the ordinary pathwidth: admitting the $\Omega(\log n \cdot tw(G))$ lower bound for some infinite class of graphs (the constant at the Ω may depend on d). If this conjecture is confirmed, an XP lower bound in terms treewidth will follow for d-SMNBPs for each constant d. The second research direction is to 'upgrade' the lower bound for d-SMNBPs to a lower bound for a more 'mainstream' model. As a first step towards this direction, we pose open questions as to whether the proposed lower bound holds for d-MNBPs (without the 'separable' assumption) and whether $\psi(G)$ for G with 2-pathwidth at least k requires an exponential in k size representation by a conjunction of two OBDDs. The last question aims to investigate the splitting power in the non-monotone case. The non-monotone splitting power of even read-twice branching programs is greatly enhanced by the existence of inconsistent paths. This turns the splitting of the set of clauses (which is, essentially, the case for monotone branching programs) into intersection of the sets of satisfying assignments which can be very chaotic and way harder to grasp. In fact, this is exactly the reason why our argument for the monotone lower bound fails in the non-monotone case.

1.3 Motivation

Representation of CNFs of bounded treewidth in terms of 'weak' classes of Boolean circuits is being actively studied in the area of knowledge compilation. A well known fact is the existence of a 'watershed' between DNNF (Decomposable Negation Normal Forms) based models and those based on read-once branching programs in the sense that the former have FPT sized representation of CNFs of bounded primal treewidth while the latter do not in general. So, it is interesting to see if the 'border' can be crossed by branching programs through a slightly increased repetition. In this paper we show that the answer is positive for CNFs with bounded number of occurrences of each variable but in general the question remains open.

From the graph theoretical perspective, the new parameters, the result for bounded degree graphs, and several open questions contribute to the area of graph sparsity as per the landmark book [11]. The book discusses several variants of vertex and edge colouring where monochromatic components are sparse. The graphs whose edges can be coloured in c colours so that the pathwdith of each monochromatic component is at most k are graphs with c-pathwidth k. In fact, the definition of c-pathwidth allows edges to have several colours (that is, to participate in several monochromatic components).

From the perspective of circuit complexity, the lower bound proposed in this paper can be seen as a lower bound for a representation of a 'simple' circuit by a restricted model (this point of view is stated in [18] in the context of FBDDs). In case of the concrete lower bound for $\psi(K_n)$, the circuit is so simple that it can be represented by the very same restricted model with a slightly increased repetition. From the perspective of future research, the most intriguing question is whether the scalable lower bound for $\psi(K_n)$ holds without the monotonicity assumption. The positive answer will mean a significant breakthrough breaking the polylogarithmic repetition barrier for branching programs (see the second paragraph in the next subsection for the related background). On the other hand, the negative answer will also mean a significant insight as to how non-monotonicity 'beats' monotonicity in this particular context.

1.4 Related work

An FPT upper bound for DNNFs parameterized by treewidth has been obtained in [6]. Subsequent research resulted in refinement of the upper bound to several restricted DNNF classes such as Decision DNNFs [12]. On the side of syntactic read-once branching programs, 1-NBP requires, in general, XP size parameterized by CNFs primal treewidth [14]. The result of [14] also holds for c-OBDDs for a constant c [15] (even if they are generalized to being non-deterministic and semantic). However, the querying pattern of c-OBDDs is quite restrictive and allows establishing lower bounds that (to the best of our knowledge) are not currently known for more general models of branching programs.

For (syntactic) k-NBPs exponential lower bounds are known for $k = O(\log n)$ [5]. With the extra assumption that the branching programs are deterministic and oblivious, k can be increased to $O(\log^2 n)$ [2]. We are not aware of exponential lower bounds for higher values of k even if the oblivious branching program is further restricted to k-IBDDD [3]. In fact, we are not aware for such lower bounds even if we restrict a k-IBDD to be just a conjunction of k OBDDs. On the other hand, for k-OBDDs, exponential lower bounds are known for $k = o(n/logn)$ (Corollary 7.5.10 of [18]).

Our definition of monotone NBPs (disallowing negative literals as labels on the edges) is as in [9] and [13]. It is known that there are problems in monotone NP require monotone circuits of exponential size [1] and there are problems in monotone P requiring monotone formulas of exponential size [17]. The latter lower bound implies an exponential lower bound for monotone switching and rectifier networks, that is monotone NBPs with unbounded repetition [13].

Several new structural graph parameters have been introduced in the recent years, e.g. [4] and [10]. The parameters introduced in this paper can be seen as a contribution to this trend.

1.5 Structure of the paper

Section 2 introduces the necessary background. In Section 3 we define the new graph parameters and upper bound them by a function of treewidth and max-degree of the considered graph. Next, in Section 4, we use the upper bound to establish an upper bound for a conjunction of two OBDDs. Finally, in Section 5, we prove a lower bound for the separable monotone branching programs of bounded repetition. Proofs of some statements of Section 5 are postponed to Section 6.

2 Preliminaries

2.1 Models of Boolean functions

A literal is a Boolean variable or its negation. In this paper when we use a set of literals, we mean a *proper* set of literals where a variable cannot occur along with its negation. If a variable x occurs in a set S of literals, it can occur *positively*, if $x \in S$ or *negatively*, if $\neg x \in S$. We denote by $Var(S)$ the set of variables occurring in S. We also call S an *assignment* to $Var(S)$.

A *CNF* φ is a set of *clauses* and each clause is just a set of literals. We denote by $Var(\varphi)$ the set of all variables occurring in the clauses. A set S of literals *satisfies* a clause C is $S \cap C \neq \emptyset$. S satisfies a set of clauses if it satisfies each clause of the set. A *satisfying assignment* of a CNF φ is an assignment to $Var(\varphi)$ satisfying φ.

For a Boolean function F, we denote by $Var(F)$ the set of variables of this function. An assignment S of $Var(F)$ is a satisfying assignment for F if F is $true$ on the tuple where each variable occurring positively in S is assigned with $true$ and each variable occurring negatively in S is assigned with $false$.

Definition 1 (NBPs) *A* non-deterministic branching program *(NBP) B is a directed acyclic graph (DAG), multi-edges allowed, with one source and one sink some edges of which are labelled with literals. We denote by $Var(B)$ the set of all variables whose literals label the edges of B.*

A (directed) path P of B is consistent *if its labels do not include a variable along with its negation. For a consistent path P, we denote by $A(P)$ the set of literals labelling the edges of P.*

A satisfying assignment *of B is a set S of literals with $Var(S) = Var(B)$ and such that there is a consistent source-sink path P with $A(P) \subseteq S$. In this case, we sometimes say that P* carries *S.*

We say that B represents *a CNF φ (respectively, a Boolean function F) if the set of satisfying assignments of B is the same as that of φ (respectively, of F).*

We denote by $|B|$ the number of edges in B.

Remark 1 *There is no point to have two unlabelled edges between the given pair of vertices x and y or two edges labelled with the same literal. Therefore, we* *can assume that the number of multiple edges between the given pair of vertices is at most $2n + 1$, the total number of literals plus possibly one unlabelled edge. That is, $|B|$ and $|V(B)|$ (the number of vertices of B) are polynomially related. Therefore, for the purposes of this paper, we can use either measure. However, for the upper bound in Section 4, we use $|V(B)|$ simply because the bound is based on an existing upper bound that also uses $|V(B)|$. On the other hand, the lower bound in Section 5 is stated for $|B|$. The reason is, again, pure convenience: the lower bound is proved for an auxiliary branching program having at most 3 times more edges than the original one, while the number or vertices can grow quadratically because of subdivision of edges. Therefore, the use of $|B|$ preserves the asymptotical lower bound for the original branching program.*

Definition 2 (d-NBPs) *A* syntactic read-d-times *NBP (d-NBP) is an NBP B where on each path P and each variable $x \in Var(B)$, the number of occurrences of x as a label of an edge of P is at most d.*

Definition 3 (Monotone and separable branching programs) *An NBP B is* monotone *if negative literals do not occur as labels of the edges of B. An d-NBP B is* separable *if each source-sink path P can be partitioned into at most d edge-disjoint* read-once *subpaths. For example, if the sequence of variables queried along a path is x_1, x_3, x_1, x_2 then this path can be partitioned into two read-once subpath whose edges query variables x_1, x_3 and x_1, x_2, respectively. However, if the sequence is x_1, x_3, x_1, x_2, x_2 then such a partition is not possible.*

We abbreviate a monotone k-NBP as k-MNBP and a monotone separable k-NBP as k-SMNBP.

Remark 2 *The querying constraint imposed by separable d-NBPs is weaker than that of d-IBDD [3]. In particular, separable d-NBPs do not place any constraints on specific orders of querying variables within read-once fragments.*

Definition 4 (OBDDs) *An ordered binary decision diagram B is a DAG, multiple edges allowed, with a single source and two sinks, one labelled with $True$, the other labelled with $False$. Each non-sink node has two outgoing edges labelled with opposite literals of the same variable. The labelling of the edges is* read-once*: for each path P of B there are no two different edges labelled by literals of the same variable. The labelling is also* oblivious*: there is a permutation π of variables (treated as a linear order) : for each path P whenever a literal of y occurs on P after a literal of x, it holds that y occurs after x in π.*

We denote by $Var(B)$ the set of variables whose literals label the edges of B. We denote by $A(P)$ the set of literals labelling the edges of a path P of B. The function $F(B)$ represented by B is a function whose set of variables is $Var(B)$ and the set of satisfying assignments consists of precisely those S such that there is a path P from the source to the $True$ sink such that $A(P) \subseteq S$.

Definition 5 *Let $B_1, \dots, B_q$ be OBDDs. The function $F = F(B_1) \wedge \dots \wedge F(B_q)$ is called the* conjunction *of $B_1, \dots, B_q$*

Remark 3 *The conjunction of OBDDs $B_1, \dots, B_q$ can be easily represented as q-NBP as follows. Transform each B_i into a 1-NBP B'_i by removal of the $False$ sink and all the nodes from which the $True$ sink cannot be reached. Then for each $1 \leq i \leq q-1$ identify the sink of B_i with the source of B_{i+1}. The same 'chaining' approach but with a slightly more tedious implementation can be used to demonstrate that the conjunction of q OBDDs can be represented as a q-IBDD, a restricted class of deterministic read q times branching programs.*

2.2 Graphs and their structural parameters

We use a standard terminology related to graphs as in e.g. [7]. In particular, we denote by $G[S]$ the subgraph of G induced by $S \subseteq V(G)$.

Definition 6 (Treewidth and pathwidth) *A* tree decomposition *of a graph G is a pair $(T, \mathbf{B})$ where T is a tree and $\mathbf{B}$ is a set of bags B_x corresponding to the nodes x of T. Each bag is a subset of $V(G)$ and the following conditions must be met: (i)* union, *that is $\bigcup_{x \in V(T)} B_x = V(G)$, (ii)* containment, *that is for each $e \in E(G)$ there is $x \in V(T)$ such that $e \subseteq B_x$, and (iii)* connectedness, *that is for each $u \in V(G)$, the set $\{x | u \in B_x\}$ induces a connected subgraph of T.*

If T is a path then $(T, \mathbf{B})$ is called a path decomposition *of G.*

The width of $(T, \mathbf{B})$ is the size of the largest bag minus one. The treewidth *of G, denoted by $tw(G)$ is the smallest width of a tree decomposition of G. The* pathwidth *of G, denoted by $pw(G)$ is the smallest width of a path decomposition of G.*

We conclude this section with three facts about treewidth and pathwidth along with literature references for relevant proofs.

Proposition 1 *If S is a clique of G then S s a subset of a bag in every tree decomposition of G.*

Proposition 2 *A graph of treewidth k has at most nk edges.*

Proposition 3 *There is an infinite class $\mathcal{G}$ of graphs for which there is a constant α such that for each $G \in \mathcal{G}$, $pw(G) \geq \alpha \cdot tw(G) \cdot \log n$.*

Proposition 1 appears in [7] as Lemma 12.3.5, Proposition 2 appears in [16] as statement 1.10. A class as stated in Proposition 3 with an additional property that the max-degree of all graphs is 5 is provided in [14].

3 New parameters and their upper bound for graphs of bounded degree

Definition 7 *Let $d \geq 1$ be an integer and G be a graph. The d-*pathwidth *of G denoted by $pw_d(G)$is the smallest k such that there are subgraphs $G_1, \dots G_d$ of G each of pathwidth at most k and such that $G = G_1 \cup \dots \cup G_d$.*

The clique preserving *d-pathwidth denoted by $cpw_d(G)$ is defined analogously with the only extra requirement that each complete subgraph of G is a subgrpah of some G_i.*

Example 1 *A rectangular grid has* 2*-pathwidth* 1*. Indeed, let one subgraph be induced by all the 'horizontal' edges and the other subgraph be induced by all the 'vertical' edges. This way the grid is represented as the union of two subgraphs each connected component of each subgraph is a path.*

Example 2 *A tree has* 2*-pathwidth* 1*. Indeed, let T be a tree. Pick an arbitrary node r of T and let it be the root. Then the edges are naturally divided into layers. The edges between the root and its children are of layer one. the edges between the children of the root and their children and layer* 2 *and so on. Let G_1 and G_2 be the subgraphs of G induced by the edges of the odd and even layers, respectively. Then each connected component of each G_i is a star and hence both subgraph have pathwidth* 1*. This approach is demonstrated in Figure 1.*

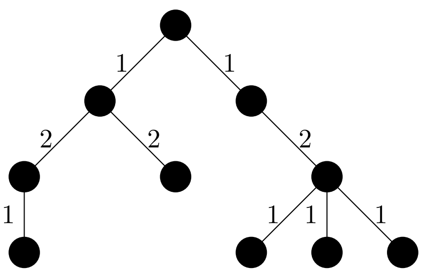

Figure 1: A tree as the union of two subgraphs of pathwidth 1. The edges of one subgraph are labelled with 1, the edges of the other subgraph are labelled with 2.

The main question studied in this section is the following: can $pw_d(G)$ and $cpw_d(G)$ be upper bounded by a function of $tw(G)$, the treewidth of G for any constant d? For graphs of a bounded degree the answer is positive already for $d = 2$, for graphs in general we will conjecture that this is not the case in a strong sense.

In order to address the case of a bounded degree we need the notion of a tree partition width.

Definition 8 *Tree-partition decomposition of a graph G is a pair $(T, \mathbf{B})$ where T is a forest and $\mathbf{B}$ is a set of bags corresponding to the nodes of T that constitute a* partition *of $V(G)$. Furthermore, let t_1, t_2 be two distinct nodes of T. Then $B(t_1)$ and $B(t_2)$ are adjacent in G (meaning a vertex of $B(t_1)$ is adjacent to a vertex of $B(t_2)$) if and only if t_1 and t_2 are adjacent in T. The width of $(T, \mathbf{B})$ is the largest size of a bag. The tree-partition width of G denoted by $tpw(G)$ is the smallest width of a tree-partition decomposition of G.*

The tree partition width of G can be linearly upper bounded by the product of $tw(G)$ and $\Delta(G)$ [19].

Theorem 1 *There is a constant γ such that for each graph G with at least one edge, $tpw(G) \leq \gamma\Delta(G)tw(G)$, where $\Delta(G)$ is the max-degree of G.*

We are going to show that $cpw_2(G) \leq 2 \cdot tpw(G) - 1$. Combined with Theorem 1, this implies that $cpw_2(G) = O(\Delta(G)(tw(G))$. In order to upper bound $cpw_2(G)$ in terms of $tpw(G)$ we turn the witnessing tree-partition decomposition for $tpw(G)$ into a tree decomposition by leaving the same underlying tree with arbitrarily identified root and by adding the vertices of the bag of the parent to each non-root bag. The set of nodes of the underlying rooted tree are naturally partitioned into layers. We enumerate the layers from the top to the bottom. Then we create two graphs one induced by the union of bags of even layers and one induced the union of bags at odd layers. We then show that these two graphs witness the bounded $cpw_2(G)$.

Example 2 and Figure 1 illustrate this approach in case G is a tree. For a larger tree-partition width, we cannot consider graphs induced solely by the edges between the bags of a witnessing tree-partition decomposition as we also need to consider edges between vertices that get to the same bag. Therefore, the sketch provided in the previous paragraph differs from the one in Example 2 even though both are based on the same idea. A formal description is provided in the theorem below.

Theorem 2 $cpw_2(G) \leq 2tpw(G) - 1$. *In particular, it follows from Theorem 1 that $cpw_2(G) \leq 2\gamma \cdot \Delta(G) \cdot tw(G)$.*

Proof. Let $(T, \mathbf{B})$ be a tree-partition decomposition of width $tpw(G)$. Turn T into a rooted tree by arbitrarily picking a node of T and calling it $root(T)$. For each non-root node $t \in V(T)$, let $p(t)$ be the parent of t. Let $(T, \mathbf{B}^*)$ be a pair where $\mathbf{B}^*$ is a set of bags associated with the nodes of T so that $B^*(root(T)) = B(root(T))$ and, for every other $t \in V(T)$, $B^*(t) = B(t) \cup B(p(t))$. It is not hard to see that $(T, \mathbf{B}^*)$ is a tree decomposition of G. We call it the tree decomposition *induced* by $(T, \mathbf{B})$.

Next, we partition the nodes of T into *layers* as follows: $root(T)$ is the only node of layer 1, the children of the root are the nodes of layer 2, the children of children are the nodes of layer 3 and so on. Put it differently, the layer number of a node is its distance from the root plus one. We denote by $Even(T)$ and $Odd(T)$ the sets of nodes that belong to the even and odd layers respectively. Let $V_{even} = \bigcup_{t \in Event(T)} B^*(t)$ and $V_{odd} = \bigcup_{t \in Odd(T)} B^*(t)$. We denote $G[V_{even}]$ and $G[V_{odd}]$ by G_{even} and G_{odd} and call them the *even* and *odd* subgraphs of G.

We claim that G_{even} and G_{odd} are the subgraphs witnessing $cpw_2(G) \leq 2tpw(G) - 1$. Indeed, by Proposition 1, each complete subgraph of G is a subgraph of some $G[B^*(t)]$ and hence, by construction, a subgraph of either G_{even} or G_{odd}. It remains to show that the pathwidth of G_{even} and G_{odd} is at most $2tpw(G) - 1$. We demonstrate this for G_{odd}, the proof for G_{even} is symmetric.

We construct a path decomposition of G_{odd} whose bags are $B^*(t)$ for each $t \in Odd(t)$ By construction, the size of each bag is at most $2tpw(G)$. It remains to be shown that the bags can be linearly ordered so that the bags containing each vertex of G form an interval.

Partition $Odd(T)$ into subsets $\{S_1, \dots, S_q\}$ such that two nodes get into the same subset if and only if they are siblings in T. Arbitrary order each S_i into a sequence and denote it by π_i. Let $\pi = \pi_1 + \dots + \pi_q$. With a slight abuse of notation, we treat π as a path with an edge between every pair of consecutive elements. We claim that $(\pi, \mathbf{B_0})$ where for each $t \in Odd(T)$, $B_0(t) = B^*(t)$ is a path decomposition of G_{odd}. As said in the previous paragraph, it remains to verify the connectedness property.

Let $u \in V(G_{odd})$. Let $t \in T$ such that $u \in B(t)$. If $t \in Odd(T)$ then in $(T, \mathbf{B}^*)$ u belongs to $B^*(t)$ and, possibly, to the bags of the children of t that all belong to $Even(t)$. It follows that the only bag of $\mathbf{B_0}$ containing u is $B_0(t)$ and hence the connectedness clearly holds. If $t \in Even(T)$ then in $(T, \mathbf{B}^*)$, apart from $B^*(t)$, u also belongs to the bags of the children of t that are all siblings and form some S_i. The bags of these S_i in $\mathbf{B_0}$ are the only bags containing u. By construction they form an interval. ■

As $pw_2(G) \leq cpw_2(G)$, the above upper bound holds for $pw_2(G)$ as well.

We do not know whether for a constant d, $pw_d(G)$ can be upper-bounded by a function of $tw(G)$ alone. Moreover, we are not aware of existing results supporting intuition that this might be the case. On the other hand, it is known that $tpw(G)$ cannot be upper bounded by a function of $tw(G)$ alone [19]. We therefore conjecture that the lower bound as in Proposition 3 also holds for d-pathwidth.

Conjecture 1 *For each integer $d \geq 1$ there is a constant α_d and an infinite set I_d of positive integers so that for each $k \in I$, there is an infinite class $\mathbf{G}_k$ of graphs of treewidth k such that for each $G \in \mathbf{G}_k$, $pw_d(G) \geq \alpha_d \cdot k \cdot \log n$.*

Remark 4 *It is not hard to see that the connected components of both graphs G_{even} and G_{odd}, as in the proof of Theorem 2, are subgraphs of G induced by $B(t)$ for some $t \in V(T)$ and, possibly, the respective bags of the children of t. Let P be a path consisting of vertices of such a component. As in $(T, \mathbf{B})$, two vertices in the bags of distinct children of t are not adjacent (by the properties of the tree-partition width) and the path of vertices of the same bag is of size at most $tpw(G)$, any subpath of P of length $tpw(G)+1$ contains a vertex of $B(t)$. Therefore the length of P is at most $(tpw(G)+1) \cdot tpw(G) \leq O(\triangle^2(G) \cdot tw^2(G))$ by Theorem 1. In other words the path length in each G_{even} and G_{odd} is upper bounded by a function of the max-degree and the treewidth of G. Therefore, a reasonable first step towards resolving Conjecture 1 would be to design a class of graphs in which there is no 2-colouring of edges with the length of monochromatic paths upper bounded by a function of the treewidth.*

4 Bounded treewidth and degree CNFs and conjunction of OBDDs

Throughout this section φ is a CNF and $|Var(\varphi)|$ is denoted by n.

Definition 9 *The* primal *graph G_φ has $Var(\varphi)$ as the set of vertices. Two variables are adjacent in G_φ if and only if they occur in the same clause of φ. The primal treewidth and pathwidth of φ are respective treewidth and pathwdith of G_φ and are denoted by $tw(\varphi)$ and $pw(\varphi)$, respectively.*

By analogy with graphs, we introduce the notion of degree and max-degree of variables of a CNF.

Definition 10 *The* degree *$d_\varphi(x)$ of a variable x in φ is the number of clauses where x occurs. The value $\Delta(\varphi) = max_{x \in Var(\varphi)} d_\varphi(x)$ is called the* max-degree *of φ.*

Remark 5 *For the sake of uniformity, we use the same Δ notation for the max-degree of a CNF and the max-degree of a graph. However, unlike the treewidth and pathwidth, in general, for a CNF ψ, $\Delta(\psi) \neq \Delta(G_\psi)$.*

For example consider $\psi = (x_1 \vee x_2 \vee x_3 \vee x_4)$. Clearly. the degree of each variable is 1 and hence $\Delta(\psi) = 1$. However, G_ψ is a clique of 4 vertices and hence $\Delta(G_\psi) = 3$.

In general if the largest clause size of ψ is c then $\Delta(G_\psi) \leq \Delta(\psi) \cdot (c-1)$ (a variable x can occur in at most $\Delta(\psi)$ clauses and it is adjacent in G_ψ to at most $c-1$ variables per clause). As a clause size of ψ cannot be larger than $tw(\psi)+1$, we conclude that $\Delta(G_\psi) \leq \Delta(G_\psi) \cdot tw(\psi)$ (see the proof of Theorem 4 for detailed reasoning).

Though not of direct relevance for our discussion, we also note that $\Delta(G_\psi)$ can be smaller than $\Delta(\psi)$. Indeed, consider $\psi = (x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$. For this CNF, $E(G_\psi)$ consists of a single edge connecting x_1 and x_2 and hence $\Delta(G_\psi) = 1$. On the other hand, each variable is presented in 4 different clauses and hence $\Delta(\psi) = 4$.

In this section we show φ can be represented as a conjunction of two OBDDs of size at most $2^{O(\Delta(\varphi) \cdot tw(\varphi)^2)} \cdot n$. For the proof we will use Theorem 2 (in particular, we will clarify why we need the clique preserving variant of 2-pathwidth) and the following result from [8].

Theorem 3 *A CNF ψ can be represented by an OBDD of at most $2^{pw(\psi)} \cdot |Var(\psi)|$ nodes.*

Theorem 4 *A CNF φ can be represented as the conjunction of two OBDDs of at most $2^{2\gamma \cdot \Delta(\varphi) \cdot tw(\varphi)^2} \cdot n$ nodes each where γ is the constant as in Theorem 2.*

Proof. We demonstrate the existence of two CNFs φ_1 and φ_2 such that $\varphi_1 \wedge \varphi_2 = \varphi$ and for each $i \in 1, 2$, $pw(\varphi_i) \leq 2\gamma \cdot \Delta(\varphi) \cdot tw(\varphi)^2$, where γ is a constant as in Theorem 1. Then we apply Theorem 3.

Recall that G_φ denotes the primal graph of φ. As in the primal graph clauses turn into cliques, and each clique, in turn, is a subset of some bag by Proposition 1, no clause can be of size larger than $tw(\varphi)+1$. Thus each variable of φ can be in the same clause with at most $\Delta(\varphi) \cdot tw(\varphi)$ other variables and hence $\Delta(G_\varphi) \leq \Delta(\varphi) \cdot tw(\varphi)$.

By Theorem 2, there are two graphs G_1 and G_2 such that $G_1 \cup G_2 = G_\varphi$, each clique of G is a subgraph of some G_i and the pathwidth of each G_i is at most $2\gamma \cdot \Delta(G_\varphi) \cdot tw(\varphi) \leq \Delta(\varphi) \cdot tw(\varphi)^2$, the last inequality follows from the previous paragraph.

Create CNFs φ_1 and φ_2 as follows. For each clause C of φ, if the clique induced by C is a subgraph of G_1, let C be a clause of φ_1 otherwise let C be a clause of φ_2.

By construction, $\varphi_1 \wedge \varphi_2 = \varphi$ and G_{φ_1} is a subgraph of G_1. As for each clause C, $G[C]$ is a clique, if $G[C]$ is not a subgraph of G_1 then, by clique preservation [1], $G[C]$ is a subgraph of G_2. Therefore G_{φ_2} is a subgraph of G_2. We conclude that both $pw(\varphi_1)$ and $pw(\varphi_2)$ are at most $2\gamma \cdot \Delta(\varphi) \cdot tw(\varphi)^2$. ■

5 Lower bounds depending on d-pathwidth

Recall from the Preliminaries section that a d-MNBP Z is *separable*, abbreviated as d-SMNBP, if every source-sink path of Z can be partitioned into at most d read-once subpaths.

In this section we prove a lower bound for d-SMNBPs representing a class of CNFs defined below. The lower bound is exponential is terms of d-pathwidth of the primal graphs of these CNFs.

Definition 11 *Let G be a graph without isolated vertices. The CNF $\psi(G)$ corresponding to G has $V(G) \cup E(G)$ as the set of variables. The variables of $V(G)$ and $E(G)$ are, respectively, the* vertex *and* edge *variables. The clauses correspond to $E(G)$. In particular for each edge $e = \{u, v\}$ of G, the corresponding clause is $(u \vee e \vee v)$.*

Note that each e occurs as a variable only in the clause corresponding to e. The role of the edge variables is *padding* that allows any assignment to the vertex variables to be extended to a satisfying assignment of $\psi(G)$. The primal graph H of $\psi(G)$ is obtained from G by introducing an individual vertex for each $e = \{u, v\}$ and making it adjacent to vertices u and v.

Proposition 4 *For each positive integer d, $pw_d(G) \leq pw_d(H) \leq pw_d(G) + 1$.*

Proof. As G is a subgraph of H, the first inequality is immediate. For the other inequality, let $G_1, \dots, G_d$ be subgraphs of G of pathwidth at most $pw_d(G)$ such that$G_1 \cup \dots \cup G_d = G$. We transform each G_i into H_i as follows. For each $\{u, v\} \in E(G_i)$, add the unique new vertex x whose neighbours are u and v along with the edges connecting x to u and v. As each $\{u, v\} \in E(G)$ is an edge

[1]this is why the clique preserving variant of 2-pathwidth is needed!

of some G_i, the new vertex x whose neighbours are u and v belongs to $V(H)$ and the two edges adjacent to x belong to $E(H)$. We conclude that $H_1 \cup \dots \cup H_d = H$. It remains to show that for each $i \in \{1, \dots, d\}$, $pw(H_i) \leq pw(G_i) + 1$.

Let $(P, \mathbf{B})$ be a path decomposition of G_i having the smallest possible width. For each $x \in V(P)$ let $r(x) = |E(G_i[B_x])|$. Put it differently, $r(x)$ is the number of edges of G_i between vertices of B_x. Form a new path P' by replacing each x with a sequence of $r(x)$ nodes and let $B(x)$ be the bag of each node. Thus the bags $B(x)$ are now in a bijective correspondence with $E(G_i[B_x])$. Now, add to each bag $B(x)$ the new vertex of H_i corresponding to the edge of $E(G_i[B_x])$ that corresponds to this bag. This way all the vertices of H_i are accommodated and each bag becomes larger by at most one element. Each new vertex belongs to exactly one bag and for each old vertex u the subpath of the nodes of P whose bags contain u may become longer in P' but is still a subpath. Hence, we have obtained a path decomposition of H_i of width at most $pw(G_i) + 1$. ■

In light of Proposition 4, we use $pw_d(G)$ rather than $pw_d(H)$ in the main theorem of this section and the lower bound in terms of $pw_d(H)$ readily follows.

Theorem 5 *Let $d, k \geq 1$ be integers. Let G be a graph with $pw_d(G) \geq k$. Let B be a d-SMNBP with representing $\psi(G)$. Then $|B| \geq \Omega(\beta^k)$ where $\beta = (8/7)^{1/12}$. In particular, subject to Conjecture 1 being true, for every constant d, there is no FPT-sized d-SMNBP representation of CNFs of bounded treewidth.*

Before proving Theorem 5, we demonstrate its application by proving a lower bound for CNFs $\psi(K_n)$ that is tight in the sense described below.

Theorem 6 *Let a and b be positive constants such that $a + b < 1$. Then, for a sufficiently large n, any n^a-SMNBP representing K_n has size at least $\Omega(\beta^{n^b})$ where $\beta = (8/7)^{1/12}$ (as in Theorem 5).*

On the other hand, $\psi(K_n)$ has an $O(n^2)$ representation as a n-SMNBP

Proof. Represent K_n as the union of graphs $G_1, \dots, G_q$ so that $q \leq n^a$. Then at least one G_i will have at least $\binom{n}{2}/n^a$ edges. For a sufficiently large n, $\binom{n}{2}/n^a > n \cdot n^b$. As $pw(G_i) \geq tw(G_i)$ by Proposition 2, $pw(G_i) \geq n^b$. We conclude that the $pw_{n^a}(K_n) > n^b$. The lower bound as specified in the statement immediately follows from Theorem 5.

For the upper bound, represent K_n as the union of n stars $K_{1,n-1}$, represent each copy of $K_{1,n-1}$ as 1-MNBP, the resulting n-SMNBP is just their conjunction. It remains to show how to represent $K_{1,n-1}$ as a 1-MNBP of size $O(n)$.

Let $v_0, \dots v_{n-1}$ be the vertices of $K_{1,n-1}$, v_0 being the centre. Let $e_1, \dots, e_{n-1}$ be the edges connecting v_0 to $v_1, \dots, v_{n-1}$ respectively. That is $\psi(K_{1,n-1}) = (v_0 \vee e_1 \vee v_1) \wedge \dots \wedge (v_0 \vee e_{n-1} \vee v_{n-1})$

Let B' be an 1-MNBP with vertices $x_0, \dots, x_{n-1}$ with x_0 being the source and x_{n-1} being the sink. Introduce an edge from x_0 to x_{n-1} and label it with v_0. Then for each $1 \leq i \leq n-1$ introduce a pair of parallel edges between x_{i-1} and x_i label one of them with v_i and the other with e_i. Figure 2 illustrates this construction for $K_{1,4}$, A direct inspection shows that B' represents $\psi(K_{1,n-1})$. ■

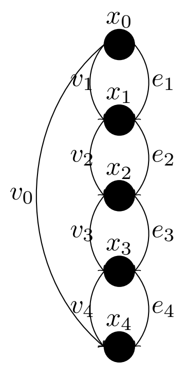

Figure 2: A 1-MNBP for $K_{1,4}$

In order to prove Theorem 5, we introduce a number of auxiliary statements. We prove of several of these statements using additional auxiliary statements that are not of direct relevance for the main line reasoning. Therefore, for the sake of readability, we postpone the proofs to Section 6.

First of all, we introduce a restricted version of d-SMNBP called d-SMNBP *with yardsticks*. We then show that the version with yardsticks simulates the d-SMNBP with only a linear increase in the number of edges. Then we prove Theorem 5 under assumption that the underlying d-SMNBP is with yardsticks. Theorem 5 without the assumption will immediately follow from the combination of these two statements.

Definition 12 *Let B be a d-SMNBP. We say that B* has yardsticks *if every path P has $a+1$ different vertices $u_1, \dots, u_{a+1}$, $a \leq d$ where u_1 is the source of B, u_{a+1} is the sink of B such that for every $1 \leq i \leq a$,* all *the u_i, u_{i+1} paths of B are read once. The sets $u_1, \dots, u_{a+1}$ are called the* yardsticks *(note that P may have several sets of yardsticks).*

Example 3 *Consider the 2-SMNBP on Figure 3. The variables labelling its edges are v_1, v_2, v_3. Also, x_1, x_2, x_3 are vertex names we need for further reasoning. Let P be the source sink path with edges labelled by v_1, v_2, v_1, v_2. The only way to partition P into two read-once fragments if to have the first fragment consisting of the first two edges and the second fragment consisting of the last two edges. In other words, the first and the last nodes of the first fragment are x_1 and x_3, respectively. However, there is another path between x_1 and x_3 both edges of which are labelled with v_3, that is this alternative path is not read-once. Therefore, path P cannot be assigned with yardsticks.*

This 2-SMNBP can be easily turned into one with yardsticks by subdivision of the edge (x_2, x_3). In particular, introduce a new vertex x_4 and replace (x_2, x_3) with two new edges (x_2, x_4) and (x_4, x_3). Label (x_2, x_4) with v_2. The resulting branching program represents the same function as the original one but both source-sink paths have yardsticks.

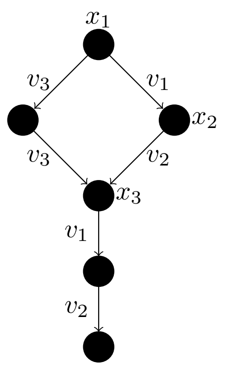

Figure 3: A 2-SMNB without yardsticks

The simulation used for the proof of Theorem 7 uses essentially the same approach.

Theorem 7 *A d-SMNBP can be simulated by an d-SMBP with yardsticks with at most three times more edges.*

Proof. Postponed to Section 6. ■

We proceed to discuss the proof of Theorem 5 under assumption that B is an d-SMNBP with yardsticks. We introduce a probability space over satisfying assignments of $\psi(G)$. We then prove that the probability of a set of satisfying assignments having a certain property is at most $(7/8)^{k/4}$. Next we prove that for each source-sink path of B, there are 3 vertices, so that the set of satisfying assignments carried by the source-sink paths passing through all these three vertices satisfies the above property. Combining the $(7/8)^{k/4}$ with the union bound implies that the number of such triples of vertices is at least $(8/7)^{k/4}$ meaning that the total number of vertices (and hence the number of edges) is lower bounded by $\Omega((8/7)^{k/12})$ as required.

To proceed, let us denote by $\mathbf{SAT}(G)$ the set of all satisfying assigments of $\psi(G)$ (recall that, by definition, the set of variables of a satisfying assignment of $\psi(G)$ is always $Var(\psi(G))$).

Definition 13 *The* Vertex-Edge *probability space of a graph G denoted by $\mathcal{VE}(G)$ is a probability distribution over $\mathbf{SAT}(G)$. The probabilities of assignments are defined as follows. Let $S \in \mathbf{SAT}(G)$. Let us call an edge $\{u, v\} \in E(G)$* free *by S if either u or v occur positively in S; otherwise, the edge is called* enforced *by S. Let $Free(S)$ be the set of free edges by S. Then $Pr_{\mathcal{VE}(G)}(S) = 2^{-(|V(G)|+|Free(S)|)}$ ($\mathcal{VE}(G)$ is the only probability space we use in this paper. Therefore, in what follows we will omit the subscript of Pr).*

It is not hard to observe that $\mathcal{VE}(G)$ is indeed a probability space. Indeed, let SV be an assignment with $Var(SV) = V(G)$. Let $\mathbf{S}(SV)$ be the set of all

$S \in \mathbf{SAT}(G)$ whose projection to the vertex variables is SV. Then $Free = Free(S)$ is the same for all $S \in \mathbf{S}(SV)$ (completely determined by SV) and hence $|\mathbf{S}| = 2^{|Free(S)|}$ (the enforced edges assigned positively, the free edges assigned arbitrarily). Then $Pr(\mathbf{S}(SV)) = 2^{-|V(G)|}$. As the set of satisfying assignments is the disjoint union of all $\mathbf{S}(SV)$, we conclude that the sum of all the probabilities is 1.

Definition 14 *Let $\mathbf{S} \subseteq \mathbf{SAT}(G)$. We say that $\mathbf{S}$* fixes *a clause C of $\psi(G)$ if there is $C' \subset C$ such that for each $S \in \mathbf{S}$, $C' \cap S \neq \emptyset$. We can also say that $\mathbf{S}$ fixes C* with *C' if a specific subset is needed in the context. We say that $\mathbf{S}$ fixes a set of clauses if $\mathbf{S}$ fixes each clause of the set.*

The use of a proper subset in Definition 14 is essential for the bottleneck argument we employ to prove a lower bound for 1-MNBP needed for the proof of Theorem 5. The bottleneck argument is, effectively, a combination of Theorem 8 and Lemma 2.

Example 4 *Consider P_4, a path of 4 vertices with v_1, v_2, v_3, v_4 being the vertices and $e_1 = \{v_1, v_2\}$, $e_2 = \{v_2, v_3\}$, $e_3 = \{v_3, v_4\}$ being the edges. That is, $\psi(P_4) = (v_1 \vee e_1 \vee v_2) \wedge (v_2 \vee e_2 \vee v_3) \wedge (v_3 \vee e_3 \vee v_4)$. Let us define a set $\mathbf{S}$ of satisfying assignments of $\psi(P_4)$ as follows.*

$\mathbf{S} = \{\{\neg v_1, \neg e_1, v_2, \neg e_2, \neg v_3, e_3, \neg v_4\},$
$\{\neg v_1, e_1, \neg v_2, e_2, \neg v_3, e_3, \neg v_4\}, \{\neg v_1, e_1, \neg v_2, \neg e_2, v_3, \neg e_3, \neg v_4\}\}$

Then $\mathbf{S}$ fixes $(v_1 \vee e_1 \vee v_2)$ with $\{e_1, v_2\}$ and $(v_3 \vee e_3 \vee v_4)$ with $\{v_3, e_3\}$. However, $\mathbf{S}$ does not fix $(v_2 \vee e_2 \vee v_3)$ as every proper subset of the clause of falsified by some assignment of $\mathbf{S}$.

A *matching* of clauses of $\psi(G)$ is a set of clauses whose corresponding edges form a matching.

Theorem 8 *For any matching M of $\psi(G)$ and any set $\mathbf{S}$ of satisfying assignments of $\psi(G)$, that fixes M, $Pr(\mathbf{S}) \leq (7/8)^{|M|}$.*

Proof. Postponed to Section 6. ■

Now we are going to show that for any source-sink path of B, there are three vertices, so that the set of satisfying assignments carried out by paths going through all these three vertices fixes a matching of size at least $k/4$ and thus Theorem 8 will imply the promised upper bound on the probability of this set of assignments.

In order to do this, we need one more definition.

Definition 15 *Let x, y be two vertices of B such that B has a path from x to y.*

- *$B(x, y)$ is the branching program obtained from B by the union of all paths from x to y along with the labels on their edges. (See Figure 4 for an example.) Accordingly, for a path P going through both x and y $P(x, y)$ is the subpath of P starting at x and ending at y (again, with the labels preserved).*

- *$\psi(x,y)$ is the set of all clauses C of $\psi(G)$ such that for each path P from x to y, $A(P)$ satisfied C.*
- *$G(x,y)$ is the subgraph of G induced by the edges corresponding to the clauses of $\psi(x,y)$.*

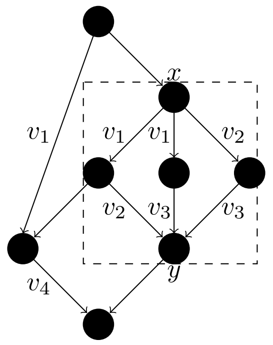

Figure 4: A branching program B. The dashed rectangle denotes $B(x,y)$

Lemma 1 *Let P be a source-sink path of B and let $u_1,\dots,u_{a+1}$ be yardsticks of P (recall that $a \leq d$). Then there is $1 \leq i \leq a$ such that $G(u_i,u_{i+1})$ is of pathwidth at least k.*

Proof. Let $\psi^* = \psi(u_1,u_2) \cup \dots \cup \psi(u_a,u_{a+1})$. We claim that $\psi^* = \psi(G)$. By construction, $\psi^* \subseteq \psi(G)$. Assume that there is a $C \in \psi(G) \setminus \psi^*$. This means that for each $1 \leq i \leq a$, there is a path P_i from u_i to u_{i+1} such that $A(P_i)$ does not satisfy C. However, $P = P_1 \cup \dots \cup P_{a+1}$ is a source-sink path and hence $A(P) = A(P_1) \cup \dots A(P_{a+1})$ must satisfy $\psi(G)$ while not satisfying C at the same time, a contradiction. [2]

It follows that $G(u_1,u_2) \cup \dots \cup G(u_a,u_{a+1}) = G$. By definition of d-pathwidth, the pathwidth of one of these graphs must be at least k. ■

In order to proceed, for a sequence L of vertices of B, we denote by $\mathbf{S}(L)$ the set of satisfying assignments of $\psi(G)$ carried by source-sink paths of B going through L. In other words, $\mathbf{S}(L)$ consists of all satisfying assignments S of $\psi(G)$ such that there is a source-sink path Q of B going through all the vertices of L such that $A(Q) \subseteq S$.

Lemma 2 *Let x,y be vertices of B such that B has a path from x to y and $B(x,y)$ is read-once. Let $r = pw(G(x,y))$. Then each x,y-path P of B has a node a such that $\mathbf{S}(L)$ fixes a matching of clauses of size at least $\lfloor r/4 \rfloor$ where $L = (x,a,y)$.*

[2] This is the argument where the monotonicity is essential. Otherwise $P_1 \cup \dots P_{a+1}$ may be an inconsistent path.

Proof. Postponed to Section 6. ■

Now, we are ready to provide a formal proof of Theorem 5, where the desired triple of vertices for each source-sink path P of B readily follows from combination of Lemma 1 and Lemma 2.

Proof of Theorem 5. In light of Theorem 7, we may assume that B is a d-SMNBP with yardsticks.

Let P be a source-sink path of B. By Lemma 1, there are consecutive yardsticks x, y of P such that $G(x, y)$ is of pathwdith at least k. Further on, by Lemma 2, there is a vertex a such that $\mathbf{S}(L)$ fixes a matching of clauses of size at least $\lfloor k/4 \rfloor$, where $L = (x, a, y)$. By Theorem 8,

$$Pr(\mathbf{S}(L)) \leq 7/8^{\lfloor k/4 \rfloor} \tag{1}$$

We call L as above a *path triple* (for P if a path needs to be specified). Let $\mathbf{L}$ be the set of all path triples. Observe that $Pr(\bigcup_{L \in \mathbf{L}} \mathbf{S}(L)) = 1$ Indeed, let S be a satisfying assignment of $\psi(G)$. Let P be a source-sink path of B such that $A(P) \subseteq S$. Then, by definition, $S \in \mathbf{S}(L)$ where L is the path triple of P. Combining the union bound with (1), we obtain, that $1 \leq \sum_{L \in \mathbf{L}} Pr(\mathbf{S}(L)) \leq |\mathbf{L}| * (7/8)^{\lfloor k/4 \rfloor}$ from where we conclude that

$$|\mathbf{L}| \geq (8/7)^{\lfloor k/4 \rfloor} \tag{2}$$

On the other hand, by construction, $|\mathbf{L}| \leq |V(B)|^3$ hence $|V(B)| \geq \Omega((8/7)^{k/12})$. As B is connected, $|B| \geq |V(B)| - 1$ and hence the statement follows. ■

Let us discuss two directions of further research.

Open Question 2 *Does Theorem 5 hold for d-MNBP (that is, without the separability assumption)?*

Since Lemma 1 is not true without the assumption that B is monotone, the above argument does not work for the non-monotone case. We believe that resolving the following open question will provide an important insight in this direction.

Open Question 3 *Is there a constant α such that for each graph G of 2-pathwidth at least k, the the size of a conjunction of two OBDDs representing $\psi(G)$ is at least α^k?*

A far fetched generalization of the last open question is whether Theorem 5 holds without the monotonicity assumption but with the extra assumption that the branching program is deterministic and oblivious. In other words, whether d-IBDD can be considered instead of d-SMNBP. In particular, is there an exponential lower bound for, say, $n^{0.01}$-IBDDs representing $\psi(K_n)$? Resolving this question positively will mean a significant breakthrough in the area of circuit complexity beating the $O(log^2 n)$ repetition barrier for oblivious branching programs (even though restricted to IBDDs). On the other hand, if the question is resolved negatively, this will result in an interesting insight as to how non-monotonicity outperforms monotonicity in this particular context.

6 Proofs omitted from Section 5

6.1 Proof of Theorem 7

We first restate the theorem.

Theorem 7 *A d-SMNBP can be simulated by an d-SMBP with yardsticks with at most three times more edges.*

Definition 16 *Let B be a d-SMNBP. A* junction vertex *of B is a vertex whose in-degree or out-degree is greater than one. Otherwise v is a* subdivision vertex. *We say that B is* subdivided *if the following two conditions hold.*

1. **Non-adjacent junctions.** *There is no edge between two junction vertices.*
2. **Subdivided literals.** *Both vertices of each edge labelled with a literal are subdivision ones.*

It is not hard to see that a d-SMNBP can be made subdivided by subdivision of each edge into three parts so that the label (if exists) is assigned to the middle part. This increases the size of the d-SMNBP at most 3 times. We are going to demonstrate that a subdivided d-SMNBP is in fact a d-SMNBP with yardsticks. In order to do this, we identify on each source-sink path a vertex we call a *pre-pivot* (the reason for this name will become clear when we provide a formal definition). The main technical statement of the proof (Theorem 9) demonstrates that each path from the source to a pre-pivot is read-once and each branching program created by the union of all paths from a pre-pivot to the sink is a $(d-1)$-SMNBP. After that Theorem 7 is proved using a simple induction.

Throughout the proof, we often regard B as a partial order relation where $u \leq v$ if and only if B has a path from u to v. The notions of minimal and maximal vertices are naturally defined in this context.

Definition 17 *A path P of B is* read-once *if it does not have two edges labelled by the same variable. A vertex v of B is* read-once *if every path from the source to v is read-once. Otherwise v is* non-read-once. *Finally v is* minimally non-read-once *if every vertex $u \neq v$ such that B has a path from u to v is read-once. We denote by $M(B)$ the set of all minimally non-read-once vertices.*

In the remaining part of the proof, for a vertex $v \in V(B)$, we denote by B_v the subgraph of B induced by v and all the vertices reachable from v, the labels of edges preserved.

The following proposition is immediate by induction on the distance from the source of B.

Proposition 5 *For every vertex $u \in V(B)$ that is not read-once there is $v \in M(B)$ such that $u \in V(B_v)$.*

Definition 18 *Let P be a source-sink path of B. Suppose that B is not read-once. The* pivot *w of P is the minimal non read-once vertex of P. The immediate predecessor of w on P is called the* pre-pivot *of P.*

Remark 6 1. *Since B is not read-once, the sink of B is not read-once. That is P has non-read-once vertices, while the source of B is read-once. Consequently, both the pivot and pre-pivot of P are well defined.*

2. *By definition, the pre-pivot of P is a read-once vertex.*

Theorem 9 *Let B be a subdivided d-SMNBP. Let P be a non-read-once source-sink path. Let v and w be, respectively, the pre-pivot and the pivot of P. Then B_v is a subdivided $d-1$-SMNBP.*

Proof. The proof is divided into two cases.

Case 1: $w \in M(B)$.

Let P_0 be a non-read-once source-w path. One of the in-coming edges of w must be labelled with a literal. Indeed, otherwise, we can take the predecessor of w on P_0 as a non-read-once vertex in contradiction to $w \in M(B)$. By the 'Subdivided literals' property of Definition 16, w is a subdivided vertex. Hence w has only one in-neighbour. As v is an in-neighbour of w, v is the only in-neigbour of w and hence, due to the absence of other incoming edges, (v, w) is labelled with a literal x.

Let Q be a source-sink path of B_v. By the 'Subdivided literals' property of Definition 16, v is a subdivided vertex. Hence the first edge of Q is (v, w). Let $P^* = P_0 \setminus \{w\} + Q$. By the previous paragraph, v is the predecessor of w on P_0 and hence P^* is a source-sink path. Hence, P^* can be partitioned into read-once fragments $P_1, \dots P_a$ occurring in the order listed with $a \leq d$. We claim that P_1 is a prefix of $P_0 \setminus \{w\}$ (not necessarily proper). Indeed, otherwise P_0 is a prefix of P_1 which is a contradiction as P_0 is not read-once. It follows that Q is a suffix of $P_2, + \dots + P_a$ and hence can be partitioned into $a - 1 \leq d - 1$ read-once subpaths. Taking into account that the property of being subdivided is preserved by taking induced subgraphs, we conclude that the theorem holds for the considered case.

Case 2: $w \notin M(B)$. Let $u \in M(B)$ be such that $w \in B_u$. By definition of the pivot, $v \notin B_u$. Consequently, in a path from u to w the predecessor of w is not v, implying that the in-degree of w is at least 2 and hence w being a junction vertex. By the properties of Definition 16 , we conclude that (v, w) is not labelled and that v is a subdivided vertex.

Now, let Q be a source-sink path of B_v. We need to demonstrate that Q can be partitioned into at most $d - 1$ read-once paths. Since v is subdivided, the first edge of Q is (v, w). Let Q_w be the suffix of Q starting at w. Since (v, w) is unlabelled, it is enough to show that Q_w can be partitioned into at most $d - 1$ read-once paths. As w is a pivot, there is a source-w non-read-once path P'_0. It is not hard to see that $P'_0 + Q_w$ is a source sink path of B. Hence $P'_0 + Q_w = P_1 + \dots + P_a$ for $a \leq d$ such that for each $1 \leq i \leq a$, P_i is a read-once path. As P_0' is not read-once, we conclude that P_1 is a prefix of P_0' and hence Q_w is a suffix of $P_2 + \ldots, P_a$ confirming the theorem.

■

Proof of Theorem 7. For each edge e of B introduce two new vertices that subdivide e into a directed path e_1, e_2, e_3. If e is labelled with a variable x, assign x to e_2. The edges e_1 and e_3 are left unlabelled. Let B^* be the resulting branching program. It is not hard to see that B^* is a subdivided d-SMNBP representing the same function as B. The theorem will immediately follow from the claim below.

Claim 1 *For each source-sink path P of B^* there is a tuple $t(P) = (v_1, \ldots, v_a)$ of vertices located on P in the order listed such that v_1 is the source, v_a is the sink, $a \leq d+1$, and for each $1 \leq i < a$, each path of B^* between v_i and v_{i+1} is read-once.*

We prove the claim by induction on d. For $d = 1$ simply associate each source-sink path with the source-sink pair. Do the same if $d > 1$ but all the paths are read-once.

So, we assume that $d > 1$ and that B^* has non read-once paths. Let P be a source-sink path. By assumption, the sink of P is non-read-once hence P has the pivot and pre-pivot. Let v be the pre-pivot of P and let Q be the suffix of P starting at v. Clearly, Q is a source-sink path of B^*_v. By Theorem 9, B^*_v is $d-1$-SMNBP.

Hence, by the induction assumption, there is a tuple $(v_2, \ldots v_a)$ of vertices of Q such that $a \leq d+1$, $v_2 = v$, v_a is a the sink and for each $2 \leq i < a$ each path of B^*_v between v_i and v_{i+1} is read-once.

Let v_1 be the source of B^*. We claim that $(v_1, \ldots, v_a)$ is the desired tuple for P. We only need to prove that for each $1 \leq i < a$ each path of B^* between v_i and v_{i+1} is read-once as the rest of the statements follow by construction. For $i = 1$ this follows from the definition of pre-pivot (see Remark 6). For $i > 1$ this follows from the previous paragraph as each path of B^* between two vertices of B^*_v is also a path of B^*_v. This proves the claim and the theorem. ■

6.2 Proof of Theorem 8

We restate the theorem first.

Theorem 8 *For any matching M of $\psi(G)$ and any set $\mathbf{S}$ of satisfying assignments of $\psi(G)$, that fixes M, $Pr(\mathbf{S}) \leq (7/8)^{|M|}$.*

The proof is based on the following idea. Let $S_1, \ldots, S_m$ be sets of variables over reals. Suppose that we want to prove that a particular value X equals $\prod_{i=1}^m \sum_{a_i \in S_i} a_i$. Then this is the same as to prove that X equals $\sum_{a_1 \in S_1, \ldots, a_m \in S_m} \prod_{i=1}^m a_i$: we simply open the brackets. In terms of probabilities, this idea can be expressed as the following statement.

Proposition 6 *Let $\mathbf{E}_1, \ldots, \mathbf{E}_m$ be events and assume that each $\mathbf{E}_i$ is the disjoint union of events $\mathbf{E}_{i,1}, \ldots, \mathbf{E}_{i,r_i}$. Let $B(\mathbf{E}_i) = \{\mathbf{E}_{i,1}, \ldots, \mathbf{E}_{i,r_i}\}$. Assume further* *that for each* $\mathbf{E'}_1 \in B(\mathbf{E}_1), \ldots, \mathbf{E'}_m \in B(\mathbf{E}_m)$*,* $Pr(\bigcap_{i=1}^m \mathbf{E'}_i) = \prod_{i=1}^m Pr(\mathbf{E'}_i)$*. Then* $Pr(\bigcap_{i=1}^m \mathbf{E}_i) = \prod_{i=1}^m Pr(\mathbf{E}_i)$*.*

In order to apply Proposition 6, we need to extend our terminology and to prove an auxiliary lemma that will allow us to easily calculate probabilities of so called *guarded* assignments.

Throughout this section when we refer to an assignment S, we mean that $Var(S) \subseteq Var(\psi(G))$. Also, S is the disjoint union of S_V and S_E where $Var(S_V) \subseteq V(G)$ and $Var(S_E) \subseteq E(G)$.

Definition 19 *Let* S *be an assignment. We denote by* $\mathbf{Ext}(S)$ *the event consisting of all the assignments that contain* S*.*

We now extend the notions of free and enforced edges as in Definition 13 to sets of literals that do not necessarily assign all of $Var(\psi(G))$.

Definition 20 *Let* S *be an assignment.*

- *Let* $e \in E(G)$*. Let* u *and* v *be the ends of* e*. We say that* e *is* guarded *(by* S*) if* $u, v \in Var(S_V)$*. The set of all guarded edges is denoted by* $Guarded(S)$*. In other words,* $Guarded(S) = \{e | e = \{u, v\} \in E(G), \{u, v\} \subseteq Var(S_V)\}$*.*
- *Let* $e \in E(G)$ *and let* u, v *be the ends of* e*. We say that* e *is* enforced *(by* S*) if both* u *and* v *occur negatively in* S_V *Otherwise,* e *is* free*. We denote by* $Enforced(S)$ *and* $Free(S)$ *the respective sets of free and enforced edges.*

Definition 21 *We say that an assignment* S *is* guarded *if* $S_E \subseteq Guarded(S)$*. We say that* S *is* valid *if all the variables of* $S_E \cap Enforced(S)$ *occur positively in* S*. Put it differently, an assignment is valid if it does not falsify any clauses.*

Lemma 3 *Let* S *be a guarded and valid assignment. Then* $Pr(\mathbf{Ext}(S)) = 2^{-(|S_V| + |Free(S) \cap S_E|)}$*.*

Proof. We assume first that $Var(S_V) = V(G)$.

Consider $S^* \in \mathbf{Ext}(S)$. By definition and our assumption, $Pr(S^*) = 2^{-(|S_v| + |Free(S^*)|)}$. As the number of free variables is completely determined by the assignment to $V(G)$, we replace $Free(S^*)$ by $Free(S)$, that is $Pr(S^*) = 2^{-(|S_V| + |Free(S)|)}$. Note that the probability of S^* is completely determined by S, that is, all the elements of $\mathbf{Ext}(S)$ have the same probability and hence $Pr(\mathbf{Ext}(S)) = 2^{-(|S_V| + |Free(S)|)} \cdot |\mathbf{Ext}(S)|$.

S^* is obtained from S by assigning variables of $E(G) \setminus S_E = (Enforced(S) \setminus S_E) \cup (Free(S) \setminus S_E)$. The elements of $Enforced(S) \setminus S_E$ must be assigned positively. The elements of $Free(S) \setminus S_E$ can be assigned arbitrarily. We conclude that $|\mathbf{Ext}(S)| = 2^{|Free(S) \setminus S_E|}$. Substituting the quantity into the formula in the end of the previous paragraph, we obtain the equality as required by the lemma.

Assume now that $S_V \subset V(G)$. Let $\mathbf{S}^*$ be the set of all $2^{|V(G) \setminus S_V|}$ extensions of S assigning the rest of vertex variables. As S is guarded, all the elements of $\mathbf{S}^*$ remain valid (otherwise, an unguarded edge variable appearing negatively, would forbid both its ends to occur negatively). Of course, all the elements of $\mathbf{S}^*$ remain guarded. Let $S^* \in \mathbf{S}^*$. By the first part of the proof, $Pr(\mathbf{Ext}(S^*)) = 2^{-(|V(G)|+|Free(S^*) \cap S^*_E|)}$. As $S^* \setminus S$ assigns vertex variables only, $S^*_E = S_E$. As all the variables of S_E are guarded in S, them being free or not is completely determined by S. Therefore, $Free(S^*) \cap S^*_E = Free(S) \cap S_E$. In other words $Pr(\mathbf{Ext}(S^*)) = 2^{-(|V(G)|+|Free(S) \cap S_E|)}$. Again, we see that this quantity is the same for all $S^* \in \mathbf{S}^*$. As $Pr(\mathbf{Ext}(S))$ is the disjoint union of $\mathbf{Ext}(S^*)$ for $S^* \in \mathbf{S}^*$, we conclude that $Pr(\mathbf{Ext}(S) = 2^{-(|V(G)|+|Free(S) \cap S_E|)} \cdot |\mathbf{S}^*| = 2^{-(|V(G)|+|Free(S) \cap S_E|)} \cdot 2^{|V(G) \setminus S_V|} = 2^{-(|S_V|+|Free(S) \cap S_E|)}$ as required.
■

Proof of Theorem 8. In order to utilize Proposition 6, we need the following claim.

Claim 2 *Let $C_1, \dots, C_q$ be a matching of clauses of $\psi(G)$. Let $S_1, \dots, S_q$ be valid assignments with $Var(S_i) = C_i$ for each $1 \leq i \leq q$. Then $Pr(\bigcap_{i=1}^q \mathbf{Ext}(S_i)) = \prod_{i=1}^q Pr(\mathbf{Ext}(S_i))$.*

Proof. As $Var(S_1), \dots, Var(S_q)$ are all disjoint by definition we can consider the assignment $S = S_1 \cup \dots \cup S_q$. It is not hard to observe that $\mathbf{Ext}(S) = \bigcap_{i=1}^q \mathbf{Ext}(S_i)$. So, we need to prove that $Pr(\mathbf{Ext}(S)) = \prod_{i=1}^q Pr(\mathbf{Ext}(S_i))$.

It is not hard to see that S is a guarded and valid assignment. By definition, S assigns $2q$ vertex variables. Let $k = |Free(S) \cap S_E|$. Then, by Lemma 3, $Pr(\mathbf{Ext}(S)) = 2^{-(2q+k)}$. On the other hand, each S_i is also a guarded and valid assignment. By Lemma 3, $Pr(\mathbf{Ext}(S_i))$ is 2^{-2} if the edge variable assigned by S_i is enforced and 2^{-3} if it is free. It is not hard to see that an edge variable assigned by S_i is free for S if and only if it is free for S_i. Therefore, there are precisely k assignments S_i for which $\mathbf{Ext}(S_i) = 2^{-3}$. That is $\prod_{i=1}^k \mathbf{Ext}(S_i) = 2^{-(2 \cdot (q-k))} \cdot 2^{-3k} = 2^{-(2q+k)}$ as required. □

Let C be a clause and C' be its proper subset. Let $CL(C, C')$ be the set of all valid assignments S with $Var(S) = C$ such that at least one variable of C' occurs positively in S. Let $\mathbf{Fix}(C, C') = \bigcup_{S \in CL(C,C')} \mathbf{Ext}(S)$. Note that this union is disjoint.

Let $C_1, \dots, C_q$ be a matching of clauses and let $C'_1, \dots, C'_q$ be their respective proper subsets. Let $\mathbf{S} \subseteq \mathbf{SAT}(G)$ be a set of assignments fixing each C_i with C'_i. Then $\mathbf{S} \subseteq \mathbf{Fix}(C_i, C'_i)$ for each $1 \leq i \leq q$. That is, $\mathbf{S} \subseteq \bigcap_{i=1}^q \mathbf{Fix}(C_i, C'_i)$. Thus theorem will follow from the combination of the following two statements.

1. $Pr(\bigcap_{i=1}^q \mathbf{Fix}(C_i, C'_i)) = \prod_{i=1}^q Pr(\mathbf{Fix}(C_i, C'_i))$.
2. $Pr(\mathbf{Fix}(C_i, C'_i)) \leq 7/8$ for each $1 \leq i \leq q$.

The first statement follows from the combination of Proposition 6 and Claim 2. For the second statement we refer to the fact that for any valid assignment S_i with $Var(S_i) = C_i$, $Pr(\mathbf{Ext}(S_i)) \geq 1/8$. At least one such S_i falsifies all the variables of C'_i and hence $\mathbf{Fix}(C_i, C'_i)$ is disjoint with $\mathbf{Ext}(S_i)$ by definition. We conclude that $Pr(\mathbf{Fix}(C_i, C'_i)) \leq 7/8$. ■

6.3 Proof of Lemma 2

We start from restating the lemma.

Lemma 2 *Let x, y be vertices of B such that B has a path from x to y and $B(x, y)$ is read-once. Let $r = pw(G(x, y))$. Then each x,y-path P of B has a node a such that $\mathbf{S}(L)$ fixes a matching of clauses of size at least $\lfloor r/4 \rfloor$ where $L = (x, a, y)$.*

First of all, it is convenient to extend the notion of a set of assignments fixing a set of clauses to the case where the assignments are partial and not necessarily over the same subset of variables.

Definition 22 *Let $\mathbf{S}$ be a family of sets of literals over subsets of $Var(\psi(G))$ (not necessarily over the same subset). We say that $\mathbf{S}$* fixes *a set $\{C_1, \dots, C_q\}$ of clauses if for each C_i there is a proper non-empty subset C'_i such that each $S \in \mathbf{S}$ satisfies all of $C_1, \dots, C_q$ and for each C_i, $S \cap C_i \subseteq C'_i$. We call $C'_1, \dots, C'_q$* witnessing subsets *of $C_1, \dots, C_q$, respectively.*

For a sequence L of vertices of B, we denote by $\mathbf{P}(L)$ the set of all paths that start at the first vertex of L, end at the last one, and go through all the intermediate ones. Accordingly, $A(\mathbf{P}(L)) = \{A(P') | P' \in \mathbf{P}(L)\}$.

In order to prove Lemma 2, we show existence of a vertex a such $A(\mathbf{P}(L))$ fixes a matching of size at least $\lfloor r/4 \rfloor$ where $L = (x, a, y)$ (as Definition 22 enables us to do so). Let us see that the statement for $\mathbf{S}(L)$ will follow. Indeed, let $C_1, \dots, C_q$ be the clauses of a matching fixed by $A(\mathbf{P}(L)$ and let $C'_1, \dots, C'_q$ be their respective witnessing subsets. Let $S^* \in \mathbf{S}(L)$. This means that there is a source-sink path P^* of B such that $A(P^*) \subseteq S^*$ and P^* goes through x, a, y. Ths means that $P^*(x, a, y) \in \mathbf{P}(x, a, y)$ and hence $A(P^*(x, a, y))$ has a non-empty intersection with all of C'_i. As $A(P^*(x, a, y)) \subseteq A(P^*) \subseteq S^*$, the same is true regarding S^*.

The advantage of considering $A(\mathbf{P}(L))$ is that the reasoning becomes 'local', confined to $B(x, y)$ and $\psi(x, y)$ rather than the whole B and $\psi(G)$.

Definition 23 *For two vertices a_1, a_2 of B let us denote by $V(a_1, a_2)$ the set of variables occurring as labels of paths between from a_1 to a_2. Put it differently, $V(a_1, a_2) = Var(B(a_1, a_2))$.*

Lemma 4 *Let b be a vertex of $B(x, y)$, let C be a clause of $\psi(x, y)$ and let $C' = C \cap V(x, b)$. Assume that $\emptyset \subset C' \subset C$. Then one of the following two statements holds.*

1. *For each path $P' \in \mathbf{P}(x, b, y)$,* [3] *$A(P')$ has a non-empty intersection with C'.*

[3]Note a slight abuse of notation: not using extra brackets for $\mathbf{P}((x, b, y))$ for the sake of better readability.

2. For each path $P' \in \mathbf{P}(x,b,y)$, $A(P')$ has a non-empty intersection with $C \setminus C'$.

Proof. Assume that the statement is not true. Then there are paths P_1 and P_2 of $\mathbf{P}(x,b,y)$ such that $A(P_1) \cap C' = \emptyset$ and $A(P_2) \cap (C \setminus C') = \emptyset$. Note that $A(P_1(x,b))$ does not intersect with C. Indeed, otherwise, by definition of b it can intersect only with C' which is impossible by definition of P_1. Further, on $A(P_2(b,y))$ does not intersect with C. Indeed, otherwise, $A(P_2(b,y))$ contains a literal u of C'. However, this literal is also contained on a path from x to b. Concatenating the former to the end of the latter, we obtain a path with a double occurrence of u in contradiction to the read-onceness of $B(x,y)$.

It follows that $P_1(x,b) + P_2(b,y)$ is an x,y-path whose set of labels does not intersect with C and hence does not satisfy C. However, this is a contradiction with our assumption that C is a clause of $\psi(x,y)$. $\blacksquare$

We utilize Lemma 4 for the proof of Lemma 2 in the following way. We consider a path P from x to y and demonstrate existence of a vertex a of P such that the premise of Lemma 4 w.r.t. $V(x,a)$ is satisfied for a matching of clauses of $\psi(x,y)$ of size at least $\lfloor pw(G(x,y))/4 \rfloor$. Then, by Lemma 4, $\mathbf{P}(x,a,y)$ fixes the matching. In order to implement this plan we need the following result that easily follows from Theorem 5 of [15].

Theorem 10 *Let H be a graph and let π be a permutation of $V(H)$. Then there is a prefix π' of π such that there is a matching of size at least $pw(H)/2$ constsing of edges with one end in π' and the other end in $\pi \setminus \pi'$.*

In order to connect Lemma 4 with Theorem 10, we introduce the definition of a witnessing permutation for P.

Definition 24 *Let P be a path of B from x to y. Let $x_1 = x, \dots, x_m = y$ be the vertices of P occurring on P in the order listed. A permutation π of $Var(\psi(x,y))$ is a* witnessing permutation *of P if $V(x,x_1) \subseteq \dots \subseteq V(x,x_m)$ are all (elements of some) prefixes of π. Put it differently, a witnessing permutation of P s created as follows. Arbitrarily order $V(x,x_2)$ (as $x = x_1$, $V(x,x_1) = \emptyset$) and let it be the initial prefix. Then, for each $3 \leq i \leq m$, if $V(x,x_{i-1}) \subset V(x,x_i)$. arbitrarily order $V(x,x_i) \setminus V(x,x_{i-1})$ and append the obtained sequence to the prefix already created. Finally, the elements of $Var(\psi(x,y)) \setminus V(x,y)$ (if any) are appended after the elements of $V(x,y)$ in an arbitrary order.*

As $V(G(x,y)) \subseteq Var(\psi(x,y))$, a witnessing permutation π of P contains a permutation π_G of $V(G(x,y))$ as a subsequence. Therefore, by Theorem 10, π has a prefix π' 'separating' a matching of clauses of size at least $pw(G)/2$. If the variables of this preifx are precisely some $V(x,x_i)$ then we are done by Lemma 4. Otherwise, we consider two cases. In the main case, π' gets in between two consecutive prefixes $V(x,x_i)$ and $V(x,x_{i+1})$ and we demonstrate that one of them separates at least half of the matching separated by π'. A formal description of this reasoning is provided in the proof below.

Proof of Lemma 2. We assume that r is a multiple of 4. Otherwise, we adjust the value of r by subtracting at most 3 from it.

Let $x_1 = x, \dots, x_m = y$ be the vertices of P occurring on P in the order listed. Let π be a witnessing permutation of P. Let π_G be the permutation of $V(G(x,y))$ induced by π (that is, for $u_1, u_2 \in V(G(x,y))$ u_1 precedes u_2 in π_G if and only if u_1 precedes u_2 in π). Let π'_G be a prefix of π_G such that there is a matching M^*_0 of $G(x,y)$ of size at least $r/2$ such that each edge of M^*_0 has one end in π'_G and one end in its complement: the existence of such a prefix is guaranteed by Theorem 10. Let π' be the prefix of π having the same last element as π'_G. Then π'_G is a subsequence of π' and $\pi_G \setminus \pi'_G$ is a subsequence of $\pi \setminus \pi'$. Let $M_0 = \{(u \vee e \vee v) | \{u,v\} \in M^*_0\}$. It follows that each clause of M_0 has a non-empty intersection with both π' and $\pi \setminus \pi'$. If the set of elements of π' is some $V(x, x_i)$, we are done by Lemma 4.

Otherwise, we consider two cases. The first, somewhat pathological case is that $V(x, x_m) \subset \pi'$. In other words, π' contains all of $Var(B(x,y))$ and some extra variables. As $A(P)$ satisfies all of $\psi(x,y)$, $V(x,y)$ intersects with all the clauses of M_0. As $V(x,y) \subset \pi'$, for each $C \in M_0$, $\emptyset \subset C \cap V(x,y) \subset C$. Hence, the result follows by Lemma 4.

In the second and the more interesting case, there is $1 \leq i < m$ such that $V(x, x_i) \subset \pi' \subset V(x, x_{i+1})$. Let M_1 be the subset of M consisting of all C such that $V(x, x_i) \cap C \neq \emptyset$. Note that by the choice of x_i, $V(x, x_i) \cap C \subseteq \pi' \cap C \subset C$ for each $C \in M_1$. Therefore, if $|M_1| \geq r/4$, we are done by Lemma 4.

Otherwise, $|M_1| \leq r/4 - 1$. Note that the label on the edge (x_i, x_{i+1}) (if exists) belongs to at most one clause C of M_0 (because the clauses of M_0 are pairwise disjoint). Let $M'_1 = M_1 \cup \{C\}$ if such a C exists. Otherwise, let $M'_1 = M_1$. Clearly, $|M'_1| \leq r/4$. Let $M_2 = M_0 \setminus M'_1$. Then $|M_2| \geq r/4$. We are going to show that for each $C \in M_2$, $\emptyset \subset V(x, x_{i+1}) \cap C \subset C$. Then the considered lemma will immediately follow from Lemma 4.

Since $\pi' \subset V(x, x_{i+1})$ and $C \cap \pi' \neq \emptyset$, $\emptyset \subset V(x, x_{i+1}) \cap C$. For the other containement, observe that $A(P(x, x_{i+1})) \cap C = \emptyset$. Indeed, $V(x, x_i) \cap C = \emptyset$ by definition and, since $A(P(x, x_i)) \subseteq V(x, x_i)$, $A(P(x, x_i)) \cap C = \emptyset$. Also, the label on (x_i, x_{i+1}), if any, is not in C by construction. On the other hand, as C is a clause of $\psi(x,y)$, $A(P) \cap C \neq \emptyset$. It follows that $A(P(x_{i+1}, y)) \cap C \neq \emptyset$. Let $w \in A(P(x_{i+1}, y)) \cap C$. Then $w \notin V(x, x_{i+1})$. Indeed, otherwise, there is a path Q from x to x_{i+1} with $w \in A(Q)$. Then $Q + P(x_{i+1}, y)$ is an x, y path of B with a double occurrence of w in contradiction to the read-onceness of $B(x,y)$. We conclude that $V(x, x_{i+1}) \cap C \subset C$ as required. ■

Acknowledgement

I would like to thank Felix Reidl for useful comments.

References

[1] Noga Alon and Ravi B. Boppana. The monotone circuit complexity of boolean functions. *Combinatorica*, 7(1):1–22, 1987.

[2] László Babai, Noam Nisan, and Mario Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *J. Comput. Syst. Sci.*, 45(2):204–232, 1992.

[3] Beate Bollig, Martin Sauerhoff, Detlef Sieling, and Ingo Wegener. Hierarchy theorems for *k*obdds and *k*ibdds. *Theor. Comput. Sci.*, 205(1-2):45–60, 1998.

[4] Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 601–612, 2020.

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

[6] Adnan Darwiche. Decomposable negation normal form. *J. ACM*, 48(4):608–647, 2001.

[7] Reinhard Diestel. *Graph Theory, 3d Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2005.

[8] Andrea Ferrara, Guoqiang Pan, and Moshe Y. Vardi. Treewidth in verification: Local vs. global. In *Logic for Programming, Artificial Intelligence, and Reasoning, 12th International Conference (LPAR)*, pages 489–503, 2005.

[9] Michelangelo Grigni and Michael Sipser. Monotone complexity. Proceedings of LMS workshop on Boolean function complexity, 1990.

[10] Dong Yeap Kang, O-joung Kwon, Torstein J. F. Strømme, and Jan Arne Telle. A width parameter useful for chordal and co-comparability graphs. *Theor. Comput. Sci.*, 704:1–17, 2017.

[11] Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012.

[12] Umut Oztok and Adnan Darwiche. On compiling CNF into decision-dnnf. In *Principles and Practice of Constraint Programming - 20th International Conference, (CP)*, pages 42–57, 2014.

[13] Alexander A. Razborov. Lower bounds for deterministic and nondeterministic branching programs. In *Fundamentals of Computation Theory, 8th International Symposium, (FCT)*, pages 47–60, 1991.

[14] Igor Razgon. On the read-once property of branching programs and cnfs of bounded treewidth. *Algorithmica*, 75(2):277–294, 2016.

[15] Igor Razgon. On oblivious branching programs with bounded repetition that cannot efficiently compute cnfs of bounded treewidth. *Theory Comput. Syst.*, 61(3):755–776, 2017.

[16] B A Reed. *Tree Width and Tangles: A New Connectivity Measure and Some Applications*, page 87–162. London Mathematical Society Lecture Note Series. Cambridge University Press, 1997.

[17] Robert Robere, Toniann Pitassi, Benjamin Rossman, and Stephen A. Cook. Exponential lower bounds for monotone span programs. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 406–415, 2016.

[18] Ingo Wegener. *Branching Programs and Binary Decision Diagrams.* SIAM Monographs on Discrete Mathematics and applications, 2000.

[19] David R. Wood. On tree-partition-width. *Eur. J. Comb.*, 30(5):1245–1253, 2009.