Technical Report AC-TR-17-012

December 2017

SAT-Encodings for Special Treewidth and Pathwidth

Neha Lodha, Sebastian Ordyniak, Stefan Szeider

This is the authors' copy of a paper that appeared in S. Gaspers and T. Walsh (Eds.): SAT 2017, LNCS 10491, pp. 429–445, 2017. DOI: 10.1007/978-3-319-66263-3_27

SAT-Encodings for Special Treewidth and Pathwidth

Neha Lodha, Sebastian Ordyniak( ), and Stefan Szeider

Algorithms and Complexity Group, Tu Wien, Vienna, Austria
{neha,ordyniak,sz}@ac.tuwien.ac.at

Abstract. Decomposition width parameters such as treewidth provide a measurement on the complexity of a graph. Finding a decomposition of smallest width is itself NP-hard but lends itself to a SAT-based solution. Previous work on treewidth, branchwidth and clique-width indicates that identifying a suitable characterization of the considered decomposition method is key for a practically feasible SAT-encoding.

In this paper we study SAT-encodings for the decomposition width parameters special treewidth and pathwidth. In both cases we develop SAT-encodings based on two different characterizations. In particular, we develop two novel characterizations for special treewidth based on partitions and elimination orderings. We empirically obtained SAT-encodings.

1 Introduction

The decomposition of graphs is a central topic in combinatorics and combinatorial optimization where various decomposition methods have been developed. Decomposition methods gives rise to a so-called width parameters that indicates how well the graph is decomposable by the considered decomposition method. For instance tree decomposition, the most famous decomposition method, gives rise to the parameter *treewidth*, where the treewidth of a graph is the smallest width over all tree decompositions. Typically, finding an optimal decomposition (i.e., one of smallest width), is an NP-hard problem, for which various exponential-time algorithms have been suggested. Previous work indicates that SAT provides a valuable practical approach for finding optimal decompositions. This approach was pioneered by Samer and Veith for treewidth [16]; their methods was further improved [2] and achieved excellent results in a recent solver challenge [8]. Heule and Szeider [11] developed the first practically feasible approach for computing the decomposition parameter *clique-width* by means of a SAT encoding, which allowed for the first time to identify the clique-width of some well-known named graphs. A SAT-encoding for the decomposition parameter *branchwidth* was suggested by Lodha et al. [14], who also showed how the encoding can be used to improve heuristically obtained branch decompositions of large graphs.

Special Treewidth and Pathwidth. In this paper we consider new SAT encodings for the decomposition parameters *special treewidth* and *pathwidth*. Special treewidth, a decomposition parameter introduced by Courcelle [6,7], is

© Springer International Publishing AG 2017
S. Gaspers and T. Walsh (Eds.): SAT 2017, LNCS 10491, pp. 429–445, 2017.
DOI: 10.1007/978-3-319-66263-3_27

closely related to the well-known decompositional parameters pathwidth and treewidth [13]. A tree decomposition of a graph G is a tree T whose nodes are labeled with sets of vertices, called *bags*, such that for each edge of G there is a bag containing both ends of the edge, and for each vertex of G, the nodes of T labeled with bags containing this vertex form a non-empty connected subtree. The width of the tree decomposition is the size of a largest bag minus 1, and the treewidth of a graph is the smallest width over all its tree decompositions. Special treewidth is defined similar to treewidth, with the additional property that T is a rooted tree, and for each vertex of G there is some root-to-leaf path in T which contains all the nodes labeled with bags containing this vertex. Pathwidth is also defined similar to treewidth, where T itself is a path. It follows from these definitions that special treewidth is in-between treewidth and pathwidth, i.e., for every graph G we have

$$\text{treewidth}(G) \;\leq\; \text{special treewidth}(G) \;\leq\; \text{pathwidth}(G).$$

The motivation for special treewidth is that it allows for more efficient model-checking algorithms for variants of Monadic Second Order Logic than treewidth, but is often smaller than pathwidth. Special treewidth has been the subject of several theoretical investigations [4,5]. Pathwidth, on the other hand, was introduced by Robertson and Seymour in the first of their famous series of papers on graph minors [15] and has since then attracted a lot of attention. The computation of special treewidth and pathwidth are NP-hard problems. For the latter this has been known [1] for long, for the former we observe that it can be deduced from known results (Theorem 1).

Characterizations of Width Parameters. Previous work on SAT encodings for treewidth, branchwidth and clique-width indicates that identifying a suitable characterization of the considered decomposition method is key for a practically feasible SAT-encoding. In fact, the standard encoding for treewidth [16] is based on the characterization of treewidth in terms of *elimination orderings*, which are linear orderings of the vertices of the decomposed graph, where after adding certain “fill-in” edges, the largest number of neighbors of a vertex ordered higher than the vertex itself, gives the width of the decomposition. For clique-width, on the other hand, no characterization based on elimination ordering is known, and the known SAT-encoding [11] uses a *partition-based* characterization, where one considers a sequence of partitions of the vertex set. A similar partition-based characterization was used for the SAT encoding of branchwidth [14]. Recently, an encoding for pathwidth and similar decompositional parameters based on the interval model of a path-decomposition has been introduced by Biedl et al. [3].

In this paper we develop and compare SAT encodings based on two characterizations of special treewidth and two characterizations for pathwidth.

Results for Special Treewidth. For special treewidth we develop a new characterization based on elimination orderings (Theorem 3), as one could expect that a characterization that is similar to the characterization successfully used for a

SAT encoding of treewidth also works well for special treewidth. We also develop a partition-based characterization which is close to the original characterization by Courcelle [6]. Our experiments show that the partition-based encoding clearly outperforms the ordering-based encoding. For instance, the former could process square grids and complete graphs being almost twice as large as the square grids and complete graphs within the reach of the latter. The partition-based encoding also beats the ordering-based encoding on many of the well-known named graphs that we consider by an order of magnitude and is competitive in running-times to the currently leading encoding for treewidth.

Results for Pathwidth. For pathwidth, there exists a well known characterization in terms of linear orderings [12] which gives rise to a natural SAT encoding, similar in spirit to the Samer-Veith encoding for treewidth [16]. However, we also considered a partition-based encoding, similar in spirit to the Heule-Szeider encoding for clique-width [11]. Our experiments indicate that the ordering-based encoding has a slight advantage on average over the partition-based encoding. However, the partition-based encoding has an extraordinary advantage on dense graphs. This encourages the development of a portfolio-based approach for SAT-encodings for pathwidth.

2 Preliminaries

2.1 Satisfiability and SAT-Encodings

We consider propositional formulas in Conjunctive Normal Form (*CNF formulas*, for short), which are conjunctions of clauses, where a clause is a disjunction of literals, and a literal is a propositional variable or a negated propositional variable. A CNF formula is *satisfiable* if its variables can be assigned true or false, such that each clause contains either a variable set to true or a negated variable set to false. The satisfiability problem (SAT) asks whether a given formula is satisfiable.

We will now introduce a few general assumptions and notation that is shared among the encodings. Namely, for our encodings we will assume that we are given an undirected graph $G = (V, E)$ and an integer ω, which represents the width that we are going to test. Moreover, we will assume that the vertices of G are numbered from 1 to n and similarly the edges are numbered from 1 to m. Details on how we used the formulas to calculate the exact width of a graph are given in Sect. 6.1. For the counting part of all our encodings we will employ the *sequential counter* approach [16] since this approach has turned out to provide the best results in our setting. To illustrate the idea behind the sequential counter consider the case that one has a variable $\mathrm{S}(v)$ for every vertex $v \in V(G)$ and one needs to restrict the number of vertices for which the variable $\mathrm{S}(v)$ is set to true to be at most some integer ω. In this case one introduces a counting variable $\#(v, j)$ for every $v \in V(G)$ and j with $1 \leq j \leq \omega$, which is true whenever there are at least j variables $\mathrm{S}(v)$ set to true in $\{\, \mathrm{S}(u) \mid 1 \leq u \leq v \,\}$. Then the following clauses ensure the semantics for the variable $\#(v, i)$ and ensure that at most ω of the variables $\mathrm{S}(v)$ are set to true. A clause $\neg\mathrm{S}(v) \vee \#(v,1)$ for every $v \in V(G)$, a clause $\neg\#(v-1,j) \vee \#(v,j)$ for every $v \in V(G)$ and j with $v > 1$ and $1 \leq j \leq \omega$, a clause $\neg\mathrm{S}(v) \vee \neg\#(v-1,j-1) \vee \#(v,j)$ for every $v \in V(G)$ and j with $v > 1$ and $1 < j \leq \omega$, and a clause $\neg\mathrm{S}(v) \vee \neg\#(v-1,\omega)$ for every $v \in V(G)$ with $v > 1$.

2.2 Graphs

We consider finite and simple undirected graphs. For basic terminology on graphs we refer to a standard text book [9]. For a graph G we denote by $V(G)$ the vertex set of G and by $E(G)$ the edge set of G. If $E \subseteq E(G)$, we denote by $G \setminus E$ the graph with vertices $V(G)$ and edges $E(G) \setminus E$.

We will often consider various forms of trees, i.e., connected acyclic graphs, as they form the backbone of tree decompositions. Let T be an undirected tree and $t \in V(T)$. We will often assume that T is rooted (in some arbitrary vertex r) and hence the parent and child relationships between its vertices are well-defined. We write T_t to denote the subtree of T rooted in t, i.e., the component of $T \setminus \{\{t,p\}\}$ containing t, where p is the parent of t in T. For a tree T, we denote by $\mathrm{h}(T)$, the *height* of T, i.e., the length of a longest path between the root and any leaf of T plus one.

2.3 Special Treewidth

To define special treewidth, it is convenient to first introduce treewidth and pathwidth and then show how to adapt the definition to obtain special treewidth.

A *tree decomposition* $\mathcal{T}$ of a graph $G = (V,E)$ is a pair (T,χ), where T is a tree and χ is a function that assigns each tree node t a set $\chi(t) \subseteq V$ of vertices such that the following conditions hold: **(T1)** for every vertex $u \in V$, there is a tree node t such that $u \in \chi(t)$, **(T2)** for every edge $\{u,v\} \in E$ there is a tree node t such that $\{u,v\} \subseteq \chi(t)$, and **(T3)** for every vertex $v \in V$, the set of tree nodes t with $v \in \chi(t)$ forms a subtree of T. The sets $\chi(t)$ for any $t \in V(T)$ are called *bags* of the decomposition $\mathcal{T}$ and $\chi(t)$ is the bag associated with the tree node t. The *width* of a tree decomposition (T,χ) is the size of a largest bag minus 1. A tree decomposition of minimum width is called *optimal*. The *treewidth* of a graph G is the width of an optimal tree decomposition of G. A *path decomposition* is a tree decomposition $\mathcal{T} = (T,\chi)$, where T is required to be a path and the *pathwidth* of a graph is the minimum width of any of its path decompositions.

A *special tree decomposition* $\mathcal{T} = (T,\chi)$ of a graph $G = (V,E)$ is a tree decomposition that is rooted at some node $r \in V(T)$ and additionally satisfies the following property [4,6]: **(ST)** for every vertex $v \in V$, the set of tree nodes t with $v \in \chi(t)$ forms a subpath of a path in T from r to a leaf. Note that **(ST)** subsumes **(T3)**, which implies that a special tree decomposition merely needs to satisfy **(T1)**, **(T2)**, and **(ST)**. The *width* of a special tree decomposition as well as the special treewidth of a graph G are defined analogously to the width of a tree decomposition and the treewidth, respectively. Figure 1 illustrates an (optimal) special tree decomposition of a graph.

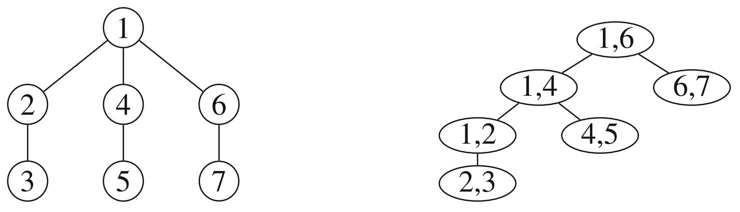

Fig. 1. A graph G (left) and an optimal (special) tree decomposition $\mathcal{T} = (T, \chi)$ of G (right).

As a prerequisite for the development of SAT-encodings for the problem, and since to the best of our knowledge this has never been explicitly stated previously, we first show that computing the special treewidth of a graph is NP-hard.

Theorem 1. *Given a graph G and an integer ω, then determining whether G has special treewidth at most ω is NP-complete.*

Proof. The problem is clearly in NP, because there is always an optimal (special) tree decomposition, where the number of nodes is at most the number of vertices in the graph. The NP-hardness follows from [10], where it was shown that pathwidth equals treewidth on the class of co-comparability graphs and moreover computing both width measures for co-comparability graphs is still NP-hard. Because special treewidth is in-between pathwidth and treewidth it equals both width measures on co-comparability graphs and its computation is therefore also NP-hard.

We remark that if ω is constant and not part of the input, then one can check in linear time whether a given graph has special treewidth at most ω (the running time depends exponentially on ω) [4]; similar results are well known to hold for treewidth and pathwidth.

2.4 Weak Partitions

A *weak partition* of a set S is a set P of nonempty subsets of S such that any two sets in P are disjoint. We denote by $\mathrm{U}(P)$ the union of all sets in P. If additionally $S = \mathrm{U}(P)$, then P is a *partition*. The elements of P are called *equivalence classes*. Let P, P' be weak partitions of S. Then P' is a *refinement* of P if $\mathrm{U}(P) \subseteq \mathrm{U}(P')$ and any two elements $x, y \in S$ that are in the same equivalence class of P' are not in distinct equivalence classes of P (this entails the case $P = P'$). Moreover, we say that P' is a *k-ary refinement* of P if additionally it holds that for every $p \in P$ there are $p_1, \dots, p_k$ in P' such that $p \subseteq \bigcup_{1 \leq i \leq k} p_i$. Intuitively, if P' is a k-ary refinement of P, then P is obtained from P' by forgetting some elements and joining up to k equivalence classes.

3 Partition-Based Approach for Special Treewidth

In this section we introduce a novel characterization of special treewidth, in terms of special derivations. The characterization is inspired by the partition-based approaches employed for branchwidth and clique-width [11,14].

3.1 Characterization: Special Derivations

Let $G = (V, E)$ be a graph. A *special derivation* $\mathcal{P}$ of G of *length* l is a sequence $(P_1, \dots, P_l)$ of weak partitions of V such that: **(SD1)** $\text{U}(P_1) = V$, **(SD2)** for every $i \in \{1, \dots, l-1\}$, P_i is a refinement of P_{i+1}, and **(SD3)** for every edge $\{u, v\} \in E$ it holds that there is a P_i and a set $p \in P_i$ such that $\{u, v\} \subseteq p$. The *width* of $\mathcal{P}$ is the maximum size of any set in $P_1 \cup \dots \cup P_l$ minus 1. We will refer to P_i as the i-th *level* of $\mathcal{P}$ and we will refer to elements in $\bigcup_{1 \leq i \leq l} P_i$ as *sets* of $\mathcal{P}$. We will show that any special tree decomposition can be transformed into a special derivation of the same width and vice verse. The following example illustrates the close connection between special tree decompositions and special derivations.

Example 1. Consider the special tree decomposition $\mathcal{T}$ given in Fig. 1. Then $\mathcal{T}$ can, e.g., be translated into the special derivation $\mathcal{P} = (P_1, \dots, P_4)$ defined by setting $P_1 = \{\{1\}, \{2, 3\}, \{4, 5\}, \{6, 7\}\}$, $P_2 = \{\{1, 2\}, \{4, 5\}, \{6, 7\}\}$, $P_3 = \{\{1, 4\}, \{6, 7\}\}$, $P_4 = \{\{1, 6\}\}$. The width of $\mathcal{T}$ is equal to the width of $\mathcal{P}$.

The following theorem shows that special derivations provide an alternative characterization of special tree decompositions. The main observation behind the proof of the equivalence between the two characterizations is that after padding the special tree decomposition such that every leaf has the same distance from the root, it holds that the weak partition on a certain level of a special derivation is given by the set of bags that are at the same distance from a leaf in a special tree decomposition and vice versa.

Theorem 2. *A graph G has a special tree decomposition of width at most ω and height at most h if and only if G has a special derivation of width at most ω and length at most h. Moreover, there is a special derivation of optimal width with length at most $|V(G)| - \omega$.*

Note that the second statement of the above theorem allows us to restrict our search to special derivations of length at most $|V(G)| - \omega$. Its proof crucially uses the observation that a restricted form of tree decompositions, so called small tree decompositions, can be shown to have height at most $|V(G)| - \omega$.

3.2 SAT-Encoding of a Special Derivation

Here we will provide our encoding for special derivations. Namely, we will construct a CNF formula $F(G, \omega, l)$ that is satisfiable if and only if G has a special derivation of width at most ω and length at most l. Because of Theorem 2 (after setting l to the value specified in the theorem) it holds that $F(G, \omega, n - \omega)$ is satisfiable if and only if G has special treewidth at most ω. To achieve this aim we first construct a formula $F(G, l)$ that is satisfiable if and only if G has a special derivation of length at most l

The formula $F(G, l)$ uses a *set variable* $\text{set}(u, v, i)$, for every $u, v \in V(G)$ and i with $u \leq v$ and $1 \leq i \leq l$. Informally, $\text{set}(u, v, i)$ is true whenever either $u \neq v$ and u and v are contained in the same set at level i of the special derivation or

$u = v$ and u is contained in some set at level i. We now describe the clauses of the formula. The following clauses ensure transitive of the relation between two vertices $u, v \in V(G)$ defined by $\mathrm{set}(u, v, i)$ for every i with $1 \leq i \leq l$.

$$\begin{aligned}
&(\neg\mathrm{set}(u,v,i) \vee \neg\mathrm{set}(u,w,i) \vee \mathrm{set}(v,w,i))\\
&\wedge(\neg\mathrm{set}(u,v,i) \vee \neg\mathrm{set}(v,w,i) \vee \mathrm{set}(u,w,i))\\
&\wedge(\neg\mathrm{set}(u,w,i) \vee \neg\mathrm{set}(v,w,i) \vee \mathrm{set}(u,v,i))\\
&\wedge(\neg\mathrm{set}(u,v,i) \vee \neg\mathrm{set}(u,u,i)) \qquad \text{for } u,v,w \in V(G), u < v < w, 1 \leq i \leq l.
\end{aligned}$$

To ensure Property **(SD1)**, we add the clause $\mathrm{set}(u, u, 1)$ for every $u \in V(G)$. The following clauses ensure **(SD2)**, i.e., P_i is a refinement of P_{i+1} for every $1 \leq i < l$.

$$\begin{aligned}
&(\neg\mathrm{set}(u,u,i+1) \vee \neg\mathrm{set}(v,v,i+1) \vee \mathrm{set}(u,v,i+1) \vee \neg\mathrm{set}(u,v,i))\\
&wedge(\mathrm{set}(u,u,i) \vee \neg\mathrm{set}(u,u,i+1)) \qquad \text{for } u,v \in V(G), u < v, 1 \leq i < l
\end{aligned}$$

Towards presenting the clauses employed to ensure **(SD3)**, we will use the following property that is easily seen to be equivalent to **(SD3)**.
(SD3') For every edge $\{u, v\} \in E$, it holds that:

- if there is an i with $1 \leq i < l$ such that $u, v \in \mathrm{U}(P_i)$ and $v \notin \mathrm{U}(P_{i+1})$, then $u, v \in p$ for some $p \in P_i$ and
- if $u, v \in \mathrm{U}(P_l)$, then $u, v \in p$ for some $p \in P_l$.

Note that **(SD3)** and **(SD3')** are equivalent because whenever there is a set $p \in P_i$ for some i with $1 \leq i \leq l$ containing two vertices u and v, then such a set also exists in every P_j for $j \geq i$ as long as $u, v \in \mathrm{U}(P_j)$. The following clauses now ensure **(SD3')** and thereby **(SD3)**.

$$\begin{aligned}
&((\neg\mathrm{set}(u,u,i) \vee \neg\mathrm{set}(v,v,i) \vee \mathrm{set}(u,u,i+1)) \vee \mathrm{set}(u,v,i))\\
&\wedge((\neg\mathrm{set}(u,u,i) \vee \neg\mathrm{set}(v,v,i) \vee \mathrm{set}(v,v,i+1)) \vee \mathrm{set}(u,v,i))\\
&\wedge((\neg\mathrm{set}(u,u,l) \vee \neg\mathrm{set}(v,v,l)) \vee \mathrm{set}(u,v,l))\\
&\qquad\qquad\qquad\qquad \text{for } e \in E(G), u,v \in e, u < v, 1 \leq i < l
\end{aligned}$$

We now ready to to extend $F(G, l)$ to the formula $F(G, \omega, l)$. We achieve this by restricting the sizes of all sets in P_i for every $1 \leq i \leq l$ to be at most $\omega + 1$, or in other words for every $v \in V(G)$ and i with $1 \leq i \leq l$, we need to restrict the number of variables $\mathrm{set}(v, u, i)$ set to true to be at most $\omega + 1$. We achieve this by using the sequential counter approach described in Subsect. 2.1. The obtained formula $F(G, l, \omega)$ contains $\mathcal{O}(n^3\omega)$ variables and $\mathcal{O}(n^4 + mn^3)$ clauses.

4 Ordering-Based Approach for Special Treewidth

In this section we introduce a second characterization of special treewidth, namely special elimination orderings, inspired by elimination orderings characterizing treewidth [13].

4.1 Characterization: Special Elimination Orderings

We start by introducing elimination orderings characterizing treewidth and then show how to adapt the notion in the context of special treewidth. Towards this aim we start with a slightly non-standard definition of elimination orderings for treewidth, from which it is particularly easy to obtain our adaptation for special treewidth.

Let G be a graph with n vertices and let $\leq_S$ be a total order $(v_1, \dots, v_n)$ of the vertices of G. For two vertices u and v with $u \leq_S v$ we denote by $N_G^{\leq_S}(u, v)$ the set of all neighbors of u in G that are larger than v w.r.t. $\leq_S$. We extend this notation to sets $U \subseteq V(G)$, where $u \leq_S v$ for every $u \in U$, by setting $N_G^{\leq_S}(U, v)$ to be the set $\bigcup_{u \in U} N_G^{\leq_S}(u, v)$. We next define the sequence $G_0^{\leq_S}, \dots, G_{n-1}^{\leq_S}$ of supergraphs of G inductively as follows: We set $G_0^{\leq_S} = G$ and for every i with $1 \leq i < n$ we let $G_i^{\leq_S}$ be the graph obtained from $G_{i-1}^{\leq_S}$ after adding all edges in the set $E_i^{\leq_S}$, which is defined as follows. Let $\mathcal{C}_i^{\leq_S}$ be the set of all components of the graph $G_{i-1}[v_1, \dots, v_{i-1}, v_i]$. Then $E_i^{\leq_S}$ is the set $\{\, \{u, v\} \mid u, v \in N_{G_{i-1}}^{\leq_S}(C, v_i) \wedge C \in \mathcal{C}_i^{\leq_S} \,\}$. We call $G_{\leq_S} = G_{n-1}^{\leq_S}$ the *fill-in graph* of G w.r.t. $\leq_S$ and $G_i^{\leq_S}$ the *i-th fill-in graph* of G w.r.t. $\leq_S$. Then any total ordering $\leq_S$ gives rise to an *elimination ordering* of G and the *width* of an elimination ordering $\leq_S$ is the maximum of $\max\{\, |N_{G_{\leq_S}}^{\leq_S}(C, v_i)| \mid C \in \mathcal{C}_i^{\leq_S} \,\}$ over all i with $1 \leq i < n$. Furthermore, the *elimination width* of a graph G is the minimum width of any elimination ordering of G. It is known that the elimination width of a graph is equal to the treewidth of a graph [13].

We are now ready to show how to adapt elimination orderings for special treewidth. Informally, the crucial observation here is that because of Property **(ST)** a special tree decomposition, in contrast to a normal tree decomposition, cannot have separate branches for components that have at least one common neighbor. This property directly translates to elimination orderings in the sense that whenever two components C and C' in $\mathcal{C}_i^{\leq_S}$ share a neighbor that comes later in the ordering, they need to be handled together both for obtaining the fill-in edges as well as for determining the width of the ordering. To formalize this idea, we say that two components C and C' in $\mathcal{C}_i^{\leq_S}$ *clash* if $N_{G_{i-1}}^{\leq_S}(C, v_i) \cap N_{G_{i-1}}^{\leq_S}(C', v_i) \neq \emptyset$. Moreover, let H be the graph with vertex-set $\mathcal{C}_i^{\leq_S}$ having an edge between two vertices C and C' if and only if their associated components clash and let $\mathcal{P}_i^{\leq_S}$ be the partition of $\mathcal{C}_i^{\leq_S}$ that corresponds to the connected components of H. Then *special elimination orderings* are obtained from elimination orderings by using $\mathcal{P}_i^{\leq_S}$ instead of $\mathcal{C}_i^{\leq_S}$ to determine both the fill-in edges as well as the width of the ordering. Formally, for special elimination orderings the set $E_i^{\leq_S}$ becomes $\{\, \{u, v\} \mid u, v \in N_{G_{i-1}}^{\leq_S}(P, v_i) \wedge P \in \mathcal{P}_i^{\leq_S} \,\}$ and the width of $\leq_S$ becomes the maximum of $\max\{\, |N_{G_{\leq_S}}^{\leq_S}(P, v_i)| \mid P \in \mathcal{P}_i^{\leq_S} \,\}$ over all i with $1 \leq i < n$. We show next that special elimination orderings properly characterize special treewidth. The main ideas behind the proof of the theorem are similar to the proof showing the equivalence between eliminations orderings and treewidth [13], however, the proof is significantly more involved due to the properties of special treewidth.

Theorem 3. *A graph G has a special tree decomposition of width at most ω if and only if G has a special elimination ordering of width at most ω.*

4.2 SAT-Encoding for Special Elimination Orderings

Here we provide our encoding for special elimination orderings as introduced in the previous subsection. In particular, we will construct a CNF formula $F(G, \omega)$ that is satisfiable if and only if G has a special elimination ordering of width at most ω. Because of Theorem 3 it then holds that $F(G, \omega)$ is satisfiable if and only if G has special treewidth at most ω. Towards this aim we first construct the formula $F(G)$ that is satisfiable if and only if G has a special elimination ordering and building upon $F(G)$ we will then use cardinality constraints to obtain $F(G, \omega)$. For the definition of the formula we use the same notation as introduced in Sect. 4.1, i.e., we refer to the required elimination ordering by $\leq_S$, and use $\mathcal{C}_v^{\leq_S}$ and $\mathcal{P}_v^{\leq_S}$ to refer to the components and parts of the graph $G_{v-1}^{\leq_S}[1, \dots, v]$ (recall that we assume that the vertices of G are numbered from 1 to n).

The formula $F(G)$ uses the following variables. An *order variable* $\mathrm{o}(u, v)$ for all $u, v \in V(G)$ with $u < v$. The variable $\mathrm{o}(u, v)$ will be true if and only if $u < v$ and $u \leq_S v$. The idea behind the variable $\mathrm{o}(u, v)$ is that it can used to model the total ordering $\leq_S$ witnessing the elimination width of G by requiring that $u \leq_S v$ for arbitrary $u, v \in V(G)$ if and only if $u = v$ or $u < v$ and $u \leq_S v$ or $u > v$ and $\neg\mathrm{o}(v, u)$. In order to be able to refer to $\leq_S$ in the clauses of $F(G)$, we define the "macro" $\mathrm{o}^*(u, v)$ by setting $\mathrm{o}^*(u, v) = \texttt{true}$ if $u = v$, $\mathrm{o}^*(u, v) = \mathrm{o}(u, v)$ if $u < v$ and $\mathrm{o}^*(u, v) = \neg\mathrm{o}(v, u)$ if $u > v$. Additionally, $F(G)$ contains an *arc variable* $\mathrm{a}(u, v)$ for all $u, v \in V(G)$. The variable $\mathrm{a}(u, v)$ is true if $u \leq_S v$ and $\{u, v\} \in E(G_{\leq_S})$ and moreover it is not true if $v <_S u$. Finally, $F(G)$ has a *part variable* $\mathrm{p}(u, v)$ for all $u, v \in V(G)$. The variable $\mathrm{p}(u, v)$ is true if and only if the vertices u and v belong to the same part in $\mathcal{P}_v^{\leq_S}$. Observe that whenever a vertex u belongs to the same part as a vertex v in $\mathcal{P}_v^{\leq_S}$, then u will also be in the same part as v in $\mathcal{P}_w^{\leq_S}$ for any w with $v \leq_S w$.

We will now provide the clauses for the formula $F(G)$. The following clauses ensure that $\mathrm{o}^*(u, v)$ is a total ordering of $V(G)$ by ensuring that the relation between u and v defined by $\mathrm{o}^*(u, v)$ is transitive:

$$(\neg\mathrm{o}^*(u, v) \vee \neg\mathrm{o}^*(v, w) \vee \mathrm{o}^*(u, w)) \quad \text{for } u, v, w \in V(G) \text{ where } u,\ v, \text{ and } w \text{ are pairwise distinct.}$$

We also introduce the clause $\mathrm{a}(u, v) \vee \mathrm{a}(v, u)$ for every $\{u, v\} \in E(G)$, which ensure that at least one of $\mathrm{a}(u, v)$ or $\mathrm{a}(v, u)$ is true for every edge $\{u, v\} \in E(G)$. Towards ensuring that the ordering $\leq_S$ represented by $\mathrm{o}^*(u, v)$ is compatible with the direction of the edges given by $\mathrm{a}(u, v)$, we introduce the clause $\neg\mathrm{a}(u, v) \vee \mathrm{o}^*(u, v)$ for every $u, v \in V(G)$. Moreover, to ensure that the relation given by

$p(u,v)$ is reflexive, i.e., every vertex belongs to its own part, we introduce the clause $p(v,v)$ for every $v \in V(G)$.

The following clauses ensure that if $\mathrm{p}(u,v)$ is true, then also $\mathrm{p}(w,v)$ is true for every w that is in the same component as u in $\mathcal{C}_v^{\leq_S}$. This is achieved by enforcing that whenever a vertex w with $w \leq_S v$ is connected via an edge in $G_{\leq_S}$ to some vertex u with $\mathrm{p}(u,v)$ being true, then also $\mathrm{p}(w,v)$ is true.

$$\begin{aligned}&(\neg \mathrm{a}(u,w) \vee \neg \mathrm{p}(u,v) \vee \neg \mathrm{o}^*(w,v) \vee \mathrm{p}(w,v))\\ &\wedge(\neg \mathrm{a}(w,v) \vee \neg \mathrm{p}(u,v) \vee \neg \mathrm{o}^*(w,v) \vee \mathrm{p}(w,v))\\ &\qquad\qquad\qquad\qquad \text{for } u,w,v \in V(G) \text{ and } u \neq w \text{ and } w \neq v.\end{aligned}$$

The following clauses complete the definition of $\mathrm{p}(u,v)$ by enforcing that whenever there is a vertex u with $u \leq_S v$ that shares a neighbor x with some vertex w with $\mathrm{p}(w,v)$ being true, then also $\mathrm{p}(u,v)$ is true, as u must also be in this part.

$$\begin{aligned}&\neg \mathrm{a}(u,x) \vee \neg \mathrm{a}(w,x) \vee \neg \mathrm{p}(w,v) \vee \neg \mathrm{o}^*(u,v) \vee \mathrm{p}(u,v)\\ &\qquad\qquad\qquad\qquad \text{for } u,w,x,v \in V(G) \text{ and } u \neq w \neq x.\end{aligned}$$

The following clauses ensure that at least one of $\mathrm{a}(u,v)$ or $\mathrm{a}(v,u)$ is true for every "fill-in edge", i.e., for every edge in $E(G_{\leq_S}) \setminus E(G)$.

$$\begin{aligned}&\neg \mathrm{p}(u_1,v) \vee \neg \mathrm{p}(u_2,v) \vee \neg \mathrm{a}(u_1,w_1) \vee \neg \mathrm{a}(u_2,w_2) \vee \neg \mathrm{o}^*(v,w_1) \vee \neg \mathrm{o}^*(v,w_2)\\ &\vee \mathrm{a}(w_1,w_2) \vee \mathrm{a}(w_2,w_1) \qquad \text{for } u_1,u_2,w_1,w_2,v \in V(G) \text{ with } w_1 \neq w_2.\end{aligned}$$

This completes the construction of $F(G)$. Informally, the crucial parts to verify the correctness of the formula are that for any ordering of the vertices of G, which is defined by the setting of the ordering variables $\mathrm{o}(u,v)$, the formula ensures that whenever $\{u,v\} \in G_{\leq_S}$ then either $\mathrm{a}(u,v)$ or $\mathrm{a}(v,u)$ is true. This way the formula ensures that all edges of $G_{\leq_S}$ are considered for the definition of the part variables $\mathrm{p}(u,v)$, which in turn ensures the correctness of the formula.

We are now ready to construct the formula $F(G,\omega)$. To achieve this it only remains to restrict the sizes of the sets $N_{G_{\leq_S}}^{\leq_S}(P,v)$ to be at most ω for every $v \in V(G)$ and $P \in \mathcal{P}_v^{\leq_S}$. Indeed we need to restrict the number of vertices w satisfying the formula $\mathrm{a}(u,w) \wedge \mathrm{p}(u,v) \wedge \mathrm{o}^*(v,w)$ for every $u,v \in V(G)$. We achieve this again by using the sequential cardinality counter described in Subsect. 2.1. This concludes the description of the formula $F(G,\omega)$, which contains $\mathcal{O}(n^2\omega)$ variables and $\mathcal{O}(n^5)$ clauses.

5 SAT-Encodings for Pathwidth

In this section we introduce our characterizations and encodings for pathwidth. Namely, we first introduce an encoding for pathwidth based on the well-known vertex separation number and then provide a second encoding based on path decompositions, which can be seen as a special case of the derivation-based encoding for special treewidth.

5.1 Partition-Based Encoding for Pathwidth

In this section we provide the partition-based encoding for pathwidth. Note that since a path decomposition has no branches, and therefore the partition on every level consists merely of a single set, the partition-based characterization of pathwidth becomes much simpler than its counterpart for special treewidth. In particular, the encoding is very closely based on the characterization of pathwidth in terms of a path decomposition, which can be equivalently stated as follows. A path decomposition can be seen as a sequence $(P_1, \dots, P_\ell)$ of bags satisfying the following conditions: (**P1**) for every $v \in V(G)$ there is a bag P_i with $v \in P_i$, (**P2**) for every i with $1 \leq i < l$, if $v \in P_i$ and $v \notin P_{i+1}$, then $v \notin P_j$ for every $j > i$. We say that the vertex v has been forgotten at level $i+1$. (**P3**) for every $u, v \in V(G)$ with $\{u, v\} \in E(G)$ and every i with $1 \leq i < \ell$, it holds that if u and v have not yet been forgotten at level i but u is forgotten at level $i+1$, then u and v are contained in P_i. In the following we describe the CNF formula $F(G, \omega, \ell)$, which for a graph G and two integers ω and ℓ is satisfied if and only if G has a path decomposition of width at most ω with at most ℓ bags. Note that since path decompositions are a special case of special tree decompositions, we can bound the maximum number of bags in an optimal path decomposition by $n - \omega$ in accordance with Theorem 2. Therefore, the formula $F(G, \omega, n - \omega)$ is satisfied if and only if G has a path decomposition of width at most ω.

$F(G, \omega, \ell)$ contains the following variables for every $v \in V(G)$ and every i with $1 \leq i \leq \ell$: The *bag variable* $\mathrm{s}(v, i)$, which is true if P_i contains the vertex v, and the *forgotten variable* $\mathrm{f}(v, i)$, which is true if the vertex v has been forgotten at some step $j \leq i$. Moreover, $F(G, \omega, \ell)$ contains the following clauses:

- for every $v \in V(G)$, the clause $\neg\mathrm{f}(v, 1)$ mirroring the property that no vertex is marked forgotten at (or before) the first bag of the path decomposition,
- for every $v \in V(G)$, the clause $\mathrm{f}(u, \ell)$, which ensures Property (**P1**),
- for every $v \in V(G)$ and every i with $1 \leq i < \ell$, the clause $\neg\mathrm{s}(v, i) \vee \neg\mathrm{s}(v, i + 1) \vee \mathrm{f}(v, i)$, which ensures that if a vertex does occur in the bag at level i but not in the bag at level $i+1$, then it is marked as forgotten.
- for every $v \in V(G)$ and every i with $1 \leq i < \ell$, the clause $\neg\mathrm{f}(v, i) \vee \neg\mathrm{s}(v, i)$, which ensures that if a vertex has already been forgotten at level i, then it does not occur in the i-th bag of the path decomposition,
- for every $v \in V(G)$ and every i with $1 \leq i < \ell$, the clause $\neg\mathrm{f}(v, i) \vee \mathrm{f}(v, i+1)$, which ensures that if a vertex is forgotten at level i then it remains forgotten at any level $j > i$ (note that these clauses together with the clauses defined in the previous item ensure Property (**P2**),
- for every $u, v \in V(G)$ with $\{u, v\} \in E(G)$ and every i with $1 \leq i < \ell$, the clauses $\mathrm{f}(u, i) \vee \mathrm{f}(v, i) \vee \neg\mathrm{f}(u, i+1) \vee \mathrm{s}(u, i)$ and $\mathrm{f}(u, i) \vee \mathrm{f}(v, i) \vee \neg\mathrm{f}(u, i+1) \vee \mathrm{s}(v, i)$, which together ensure Property (**P3**).

Finally, it remains to restrict the maximum size of the set $\mathrm{s}(u, i)$ for any level i to be at most $\omega + 1$, i.e., for every level i with $1 \leq i \leq \ell$, we need to restrict the number of variables $\mathrm{s}(u, i)$ set to true to be at most $\omega + 1$. We achieve this using the sequential cardinality counter described in Subsect. 2.1. This completes the construction of the formula $F(G, \omega, \ell)$, which including the counter variables and clauses contains $\mathcal{O}(n^2\omega)$ variables and $\mathcal{O}(n^3)$ clauses.

5.2 Ordering-Based Encoding for Pathwidth

Our second encoding for pathwidth is based on the characterization of pathwidth in terms of the vertex separation number, which is defined as follows. Given a graph G, an ordering $\leq_V$ of the vertices of G, and a vertex $v \in V(G)$, we denote by $S_{\leq_V}(v)$ the set of all vertices in G that are smaller or equal to v w.r.t. $\leq_V$. Moreover, for a subset S of the vertices of G, we denote by $\delta(S)$, the set of *guards* of S in G, i.e., the set of all vertices in S that have a neighbor in $V(G)\backslash S$. Then a graph G has *vertex separation number* at most ω if and only if there is an ordering $\leq_V$ of its vertices such that $|\delta(S_{\leq_V}(v))| \leq \omega$ for every $v \in V(G)$. It is well-known that G has vertex separation number at most ω if and only if G has pathwidth at most ω [12].

We will now show how to construct the formula $F(G, \omega)$ which is satisfiable if and only if G has vertex separation number (and hence pathwidth) at most ω. Apart from the variables needed for counting (which we will introduce later), the formula $F(G, \omega)$, has an *order variable* $\mathrm{o}(u, v)$ for every $u, v \in V(G)$ with $u < v$. The variable $\mathrm{o}(u, v)$ will be true if and only if $u < v$ and $u \leq_V v$. The idea behind the variable $\mathrm{o}(u, v)$ is that it can used to model the total ordering $\leq_V$ witnessing the vertex separation number of G by requiring that $u \leq_V v$ for arbitrary $u, v \in V(G)$ if and only if $u = v$ or $u < v$ and $u \leq_V v$ or $u > v$ and $\neg\mathrm{o}(v, u)$. In order to be able to refer to $\leq_V$ in the clauses, we define the "makro" $\mathrm{o}^*(u, v)$ by setting $\mathrm{o}^*(u, v) = \texttt{true}$ if $u = v$, $\mathrm{o}^*(u, v) = \mathrm{o}(u, v)$ if $u < v$ and $\mathrm{o}^*(u, v) = \neg\mathrm{o}(v, u)$ if $u > v$. Moreover, $F(G, \omega)$ has a *guard variable* $\mathrm{c}(v, u)$ for every $u, v \in V(G)$, which is true if $u \leq_V v$ and vertex u has a neighbor vertex w such that $v \leq_V w$, i.e., vertex u contributes to the separation number for vertex v.

We will next provide the clauses for $F(G, \omega)$. Towards ensuring that $\mathrm{o}^*(u, v)$ is a total ordering of $V(G)$, it is sufficient to ensure that the relation described by $\mathrm{o}^*(u, v)$ is transitive, which is achieved by the following clauses:

$$\neg\mathrm{o}^*(u, v) \vee \neg\mathrm{o}^*(v, w) \vee \mathrm{o}^*(u, w) \qquad \text{for } u, v, w \in V(G) \text{ where } u,\ v, \text{ and } w \text{ are pairwise distinct.}$$

The next clauses provide the semantics for the variables $\mathrm{c}(v, u)$. Namely, $\mathrm{c}(v, u)$ is set to true if $u \leq_V v$ and there is an edge $\{u, w\} \in E(G)$ with $v \leq_V w$.

$$\neg\mathrm{o}^*(u, v) \vee \neg\mathrm{o}^*(v, w) \vee \mathrm{c}(v, u) \qquad \text{for } v \in V(G),\ \{u, w\} \in E(G) \text{ and } v \neq w.$$

It remains to restrict the number of guards of each vertex set $S_{\leq_V}(v)$ given by the ordering $\mathrm{o}^*(u, v)$. Using the variables $\mathrm{c}(v, u)$ this is equivalent to restricting the number of variables $\mathrm{c}(v, u)$ that are true to be at most ω for every $v \in V(G)$. Towards this aim, we again employ the sequential cardinality counter described in Subsect. 2.1. This completes the construction of the formula $F(G, \omega)$, which including the variables and clauses used for counting has $\mathcal{O}(n^2\omega)$ variables and $\mathcal{O}(n^3)$ clauses.

6 Experiments

We run the experiments on a 4-core Intel Xeon CPU E5649, 2.35 GHz, 72 GB RAM machine with Ubuntu 14.04 with each process having access to at most 8 GB RAM. For each individual SAT call we set a timeout of 1000 s and we do not impose an overall timeout for the whole process. The compilation of all encodings is implemented in C++ and we compared the performance of the encodings using the SAT solvers Minisat 2.2 (`m`), Glucose 4.0 (`g`), and MapleSAT (`a`). As benchmark instances we used the benchmark set of well-known named graphs from the literature [17] (previously also used in [11,14]) as well as uniformly generated instances like square grids and complete graphs. In the following we will refer to the two encodings introduced in Subsects. 3.2 and 5.1 as *partition-based encodings* (P) and to the encodings introduced in Subsects. 4.2 and 5.2 as *ordering-based encodings* (O). All our experimental results as well as the code for the compilation of our encodings can be found at https://github.com/nehal73/SATencoding.

6.1 Results

Our main experimental results are provided in Tables 1 and 3. Table 1 shows our results for the benchmark set of well-known named graphs from the literature. The benchmark set is a collection of well-known small to mid-sized graphs from the literature that has already been used in the comparison of encodings for other width measures such as clique-width [11] and branchwidth [14]. For each graph in the benchmark set we run our four encodings as well as, for comparison, the encoding for treewidth based on elimination orderings [16], using the three above mentioned SAT-solvers with the aim of computing the exact width of the graph. Namely, starting from width zero ($\omega = 0$) we increased ω by one as long as either the instance became satisfiable (in which case the current ω equals the width of the graph) or the SAT-call reached the timeout of 1000 s (in which case the current ω minus 1 is a lower bound for the width of the graph). If we reached a timeout, we further increased ω until the instance could be solved again within the timeout and returned satisfiable, thereby obtaining an upper bound for the width of the graph. In three cases (marked with an asterix in Table 1) we obtained the exact width using a longer timeout of 10000 s using the partition-based encoding for special treewidth. For each width parameter the obtained width of the graph (or an interval for the width giving the best possible lower bound and upper bound obtained by any encoding) is provided in the ω column of the table. Moreover, for special treewidth and pathwidth, the table contains the two columns (P) and (O), which show the best result obtained by any SAT-solver for the partition-based and ordering-based encodings, respectively. Namely, if the exact width of the graph could be determined, then the column shows the overall running-time in seconds (the sum of all SAT-calls) for the best SAT-solver, whose initial is given as a superscript. Otherwise the table shows the best possible interval that could be obtained within the timeout or "M.O." if every SAT-call resulted in a memory out.

Table 3 shows our results for square grids and complete graphs. The idea behind using square grids and complete graphs is that they represent two types of graphs with high treewidth, i.e., sparse and dense graphs. Moreover, square grids and complete graphs are also naturally well-suited to compare the encodings along the three considered width measures (i.e., pathwidth, special treewidth, and treewidth) as it is well-known that all three width measures coincide on square grids and complete graphs. Namely, the pathwidth, special treewidth, and treewidth of an $n \times n$-grid and a complete graph on n vertices is n and $n-1$, respectively. For all our encodings, the table shows the largest size of square grids and complete graphs, whose width could be determined exactly within the timeout (using any of the three considered SAT-solvers). That is, starting from $n = 1$ we called each encoding for $\omega = n - 1$ and $\omega = n$ (in the case of the $n \times n$-grid) and for $\omega = n - 1$ and $\omega = n - 2$ (in the case of the complete graph on n vertices) and increased n as long as both calls completed within the timeout.

6.2 Discussion

In the case of special treewidth, our experiments indicate that the partition-based encoding is superior to the ordering-based encoding for all of the considered instances. Namely, the partition-based encoding can solve grids and complete graphs that are almost twice as large as the ones solvable using the ordering-based encoding (Table 3). The partition-based encoding almost always beats the ordering-based encoding by at least one order of magnitude on the well-known named graphs, and it also provides better lower bounds and upper bounds for the graphs that could not be solved exactly (Table 1). Overall, the partition-based encoding can be seen as the clear winner for special treewidth, which is somewhat unexpected for two reasons: (i) the ordering-based encoding is similar in spirit to the currently leading encoding for treewidth and (ii) asymptotically, the ordering-based encoding has fewer variables and almost the same number of clauses as the partition-based encoding (Table 2). It can be observed that the partition-based encoding for special treewidth is competitive with the leading encoding for treewidth, with both encodings showing advantages on different instances.

In the case of pathwidth the difference between the two encodings is far less pronounced. Whereas the ordering-based encoding has a clear advantage on the benchmark set of well-known named graph (Table 1), although far less significant than the advantage of the partition-based encoding for special treewidth, the partition-based encoding has an extraordinary advantage on complete graphs (Table 3). We note that both encodings have asymptotically the same numbers of clauses and variables (Table 2). It seems that in general the partition-based encoding has an advantage on dense graphs, whereas the ordering-based encoding

Table 1. Results for the benchmark set of the well-known named graphs.

Instance	$\lvert V\rvert$	$\lvert E\rvert$	Special treewidth			Pathwidth			Treewidth	
			ω	O	P	ω	O	P	ω	O
Petersen	10	15	5	5.03$^{\mathrm{m}}$	**0.48**$^{\mathrm{m}}$	5	**0.28**$^{\mathrm{a}}$	0.35$^{\mathrm{m}}$	4	0.15
Goldner-Harary	11	27	4	2.53$^{\mathrm{m}}$	**0.27**$^{\mathrm{m}}$	4	0.17$^{\mathrm{m}}$	0.17$^{\mathrm{m}}$	3	0.11
Grötzsch	11	20	5	4.27$^{\mathrm{m}}$	**0.43**$^{\mathrm{m}}$	5	**0.18**$^{\mathrm{a}}$	0.36$^{\mathrm{m}}$	5	0.28
Herschel	11	18	4	3.32$^{\mathrm{m}}$	**0.34**$^{\mathrm{m}}$	4	**0.17**$^{\mathrm{a}}$	0.20$^{\mathrm{m}}$	3	0.14
Chvátal	12	24	6	11.09$^{\mathrm{m}}$	**0.92**$^{\mathrm{m}}$	6	**0.44**$^{\mathrm{g}}$	0.69$^{\mathrm{a}}$	6	0.61
Dürer	12	18	4	7.28$^{\mathrm{a}}$	**0.63**$^{\mathrm{m}}$	4	**0.13**$^{\mathrm{m}}$	0.33$^{\mathrm{m}}$	4	0.25
Franklin	12	18	5	12.30$^{\mathrm{a}}$	**1.40**$^{\mathrm{m}}$	5	**0.30**$^{\mathrm{m}}$	0.60$^{\mathrm{m}}$	4	0.30
Frucht	12	18	4	7.56$^{\mathrm{g}}$	**0.71**$^{\mathrm{m}}$	4	**0.21**$^{\mathrm{g}}$	0.31$^{\mathrm{a}}$	3	0.12
Tietze	12	18	5	11.34$^{\mathrm{m}}$	**1.26**$^{\mathrm{m}}$	5	**0.27**$^{\mathrm{m}}$	0.53$^{\mathrm{m}}$	4	0.21
Paley13	13	39	8	22.13$^{\mathrm{m}}$	**1.16**$^{\mathrm{m}}$	8	**1.02**$^{\mathrm{a}}$	1.23$^{\mathrm{m}}$	8	2.60
Poussin	15	39	6	61.07$^{\mathrm{a}}$	**1.65**$^{\mathrm{m}}$	6	**0.39**$^{\mathrm{m}}$	0.65$^{\mathrm{m}}$	6	0.37
Clebsch	16	40	9	234.28$^{\mathrm{m}}$	**13.20**$^{\mathrm{m}}$	9	25.76$^{\mathrm{a}}$	**17.17**$^{\mathrm{a}}$	8	6.30
4×4-grid	16	24	4	97.98$^{\mathrm{m}}$	**1.13**$^{\mathrm{m}}$	4	**0.22**$^{\mathrm{a}}$	0.39$^{\mathrm{m}}$	4	0.28
Hoffman	16	32	7	204.73$^{\mathrm{a}}$	**20.22**$^{\mathrm{m}}$	7	**6.30**$^{\mathrm{g}}$	8.21$^{\mathrm{m}}$	6	2.39
Shrikhande	16	48	9	234.76$^{\mathrm{m}}$	**10.42**$^{\mathrm{m}}$	9	11.78$^{\mathrm{a}}$	**8.04**$^{\mathrm{m}}$	9	131.11
Sousselier	16	27	5	127.87$^{\mathrm{m}}$	**3.33**$^{\mathrm{m}}$	5	**0.24**$^{\mathrm{m}}$	0.62$^{\mathrm{m}}$	5	0.31
Errera	17	45	6	153.83$^{\mathrm{a}}$	**2.78**$^{\mathrm{m}}$	6	**0.40**$^{\mathrm{m}}$	0.76$^{\mathrm{m}}$	6	0.49
Paley17	17	68	12	504.54$^{\mathrm{a}}$	**15.76**$^{\mathrm{m}}$	12	106.99$^{\mathrm{a}}$	**27.52**$^{\mathrm{a}}$	11	35.23
Pappus	18	27	7	912.69$^{\mathrm{a}}$	**438.24**$^{\mathrm{g}}$	7	**16.47**$^{\mathrm{g}}$	54.62$^{\mathrm{g}}$	6	160.90
Robertson	19	38	8	1082.73$^{\mathrm{a}}$	**130.26**$^{\mathrm{m}}$	8	**11.84**$^{\mathrm{g}}$	36.02$^{\mathrm{g}}$	8	307.21
Desargues	20	30	6	1349.67$^{\mathrm{m}}$	**237.57**$^{\mathrm{g}}$	6	**0.84**$^{\mathrm{m}}$	10.16$^{\mathrm{m}}$	6	324.21
Dodecahedron	20	30	6	1564.23$^{\mathrm{a}}$	**337.20**$^{\mathrm{g}}$	6	**4**$^{\mathrm{g}}$	38.52$^{\mathrm{g}}$	4–6	4–6
FlowerSnark	20	30	6	1352.67$^{\mathrm{m}}$	**201.40**$^{\mathrm{g}}$	6	**1.04**$^{\mathrm{m}}$	10.99$^{\mathrm{m}}$	6	400.06
Folkman	20	40	7	1434.93$^{\mathrm{a}}$	**130.20**$^{\mathrm{m}}$	7	**2.84**$^{\mathrm{g}}$	23.15$^{\mathrm{m}}$	6	10.87
Brinkmann	21	42	8	2548.46$^{\mathrm{m}}$	**354.62**$^{\mathrm{m}}$	8	**14.85**$^{\mathrm{g}}$	63.71$^{\mathrm{g}}$	8	593.45
Kittell	23	63	7	160.33$^{\mathrm{g}}$	**24.70**$^{\mathrm{m}}$	7	**1.05**$^{\mathrm{m}}$	8.28$^{\mathrm{m}}$	7	4.38
McGee	24	36	8*	5–8	5–8	8	**62.47**$^{\mathrm{a}}$	524.21$^{\mathrm{g}}$	5–7	5–7
Nauru	24	36	8*	5–8	5–8	8	**181.73**$^{\mathrm{a}}$	6–8	6	457.92
Holt	27	54	10*	7–10	6–10	10	**386.16**$^{\mathrm{a}}$	8–10	7–9	7–9
Watsin	50	75	3–8	M.O.	3–8	7	**76.77**$^{\mathrm{m}}$	5–7	4–7	4–7
B10Cage	70	106	2–20	M.O.	2–20	8-16	8-16	6–16	4–17	4–17
Ellingham	78	117	3–9	M.O.	3–9	6	**22.88**$^{\mathrm{m}}$	5–7	4–6	4–6

Table 2. The number of variables and clauses for our four encodings in terms of the number n of vertices, the number m of edges m, and the width ω

	sptw		pw	
	Vars	Cls	Vars	Cls
P	$\mathcal{O}(n^3\omega)$	$\mathcal{O}(n^4 + mn^3)$	$\mathcal{O}(n^2\omega)$	$\mathcal{O}(n^3)$
O	$\mathcal{O}(n^2\omega)$	$\mathcal{O}(n^5)$	$\mathcal{O}(n^2\omega)$	$\mathcal{O}(n^3)$

Table 3. Experimental results for square-grids and complete graphs: number of vertices of largest graphs solved within the timeout

Graphs	sptw		pw	
	O	P	O	P
Square grids	16	36	81	64
Complete graphs	34	76	26	123

is better suited for sparse graphs. The results seem to indicate that different encodings should be employed for different classes of graphs. This underlines the importance of developing different encodings for the same width parameter and encourages the development of a portfolio-based approach for SAT-encodings.

Moreover, we would like to mention a few general observations concerning the performance of the three SAT-solvers. Generally the differences in the performance of the three SAT-solvers were quite minor over all encodings. In particular, the conclusions drawn about the comparison of the different encodings were the same for each of the three SAT-solvers. With respect to the special treewidth encodings, it can be inferred from Table 1 that MiniSAT has the best performance for more instances than Glucose or MapleSAT. However, we observed that Glucose was the most robust among the three solvers, since there are instances that could only be solved by Glucose and all instances that could be solved by any of the solvers could also be solved by Glucose. With respect to the pathwidth encodings, the differences between the solvers is less pronounced, each having advantages on about the same number of instances.

We also conducted initial experiments on random graphs, whose results (due to space limitations) can only be found in our github repository. Namely, we tested all our encodings on random graphs with 20, 40, and 60 vertices and edge probabilities $0.1, 0.2, \ldots, 0.9$. For each setting we generated 10 random graphs and reported the average running time for each of our encodings (we used a timeout of 2000s per SAT-call as well as an overall timeout of 6 h). Our results on random graphs strongly support our conclusions reported above concerning the relative performance of the various encodings.

7 Conclusion

We compared two SAT encodings for special tree width and pathwidth respectively. For the former we introduced two novel characterizations which might be of independent interest. Based on these characterizations for special treewidth and two related characterizations for pathwidth, we developed and empirically compared SAT-encodings for the computation of special treewidth and pathwidth. Our empirical results emphasize that the performance of SAT-encodings can strongly depend on the underlying characterization. Interestingly, for special treewidth, a partition-based encoding far outperforms an ordering-based encoding, although the latter encoding is closely related to the currently leading encoding for the prominent width parameter treewidth. It is only natural to ask whether a similar partition-based approach can be fruitful for treewidth. Moreover, for pathwidth, we obtained two SAT-encodings which both perform well, each of them having an advantage on different classes of instances; thus suggests a portfolio-based approach.

Acknowledgments. The authors kindly acknowledge the support by the Austrian Science Fund (FWF, projects W1255-N23 and P-26200).

References

1. Arnborg, S., Corneil, D.G., Proskurowski, A.: Complexity of finding embeddings in a k-tree. SIAM J. Algebr. Discret. Method. **8**(2), 277–284 (1987)
2. Berg, J., Järvisalo, M.: SAT-based approaches to treewidth computation: An evaluation. In: 26th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2014, Limassol, Cyprus, 10–12 November 2014, pp. 328–335. IEEE Computer Society (2014)
3. Biedl, T., Bläsius, T., Niedermann, B., Nöllenburg, M., Prutkin, R., Rutter, I.: Using ILP/SAT to determine pathwidth, visibility representations, and other grid-based graph drawings. In: Wismath, S., Wolff, A. (eds.) GD 2013. LNCS, vol. 8242, pp. 460–471. Springer, Cham (2013). doi:10.1007/978-3-319-03841-4_40
4. Bodlaender, H.L., Kratsch, S., Kreuzen, V.J.C.: Fixed-parameter tractability and characterizations of small special treewidth. In: Brandstädt, A., Jansen, K., Reischuk, R. (eds.) WG 2013. LNCS, vol. 8165, pp. 88–99. Springer, Heidelberg (2013). doi:10.1007/978-3-642-45043-3_9
5. Bodlaender, H.L., Kratsch, S., Kreuzen, V.J., Kwon, O.J., Ok, S.: Characterizing width two for variants of treewidth (part 1). Discr. Appl. Math. **216**, 29–46 (2017)
6. Courcelle, B.: Special tree-width and the verification of monadic second-order graph properties. In: Lodaya, K., Mahajan, M. (eds) 2010 IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS, LIPIcs, Chennai, India, vol. 8, pp. 13–29. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 15–18 Dec 2010
7. Courcelle, B.: On the model-checking of monadic second-order formulas with edge set quantifications. Discr. Appl. Math. **160**(6), 866–887 (2012)
8. Dell, H., Rosamond, F.: The 1st parameterized algorithms and computational experiments challenge–track A: Treewidth. Technical report (2016). https://pacechallenge.wordpress.com/2016/09/12/here-are-the-results-of-the-1st-pace-challenge/
9. Diestel, R.: Graph Theory: Graduate Texts in Mathematics, 2nd edn. Springer Verlag, New York (2000)
10. Habib, M., Möhring, R.H.: Treewidth of cocomparability graphs and a new order-theoretic parameter. Order **1**, 47–60 (1994)
11. Heule, M., Szeider, S.: A SAT approach to clique-width. ACM Trans. Comput. Log. **16**(3), 24 (2015)
12. Kinnersley, N.G.: The vertex separation number of a graph equals its path-width. Inf. Process. Lett. **42**(6), 345–350 (1992)
13. Kloks, T.: Treewidth: Computations and Approximations. Springer Verlag, Berlin (1994)
14. Lodha, N., Ordyniak, S., Szeider, S.: A SAT approach to branchwidth. In: Creignou, N., Le Berre, D. (eds.) SAT 2016. LNCS, vol. 9710, pp. 179–195. Springer, Cham (2016). doi:10.1007/978-3-319-40970-2_12
15. Robertson, N., Seymour, P.D.: Graph minors. I. excluding a forest. J. Combin. Theory Ser. B **35**(1), 39–61 (1983)
16. Samer, M., Veith, H.: Encoding treewidth into SAT. In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 45–50. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02777-2_6
17. Weisstein, E.: MathWorld online mathematics resource (2016). http://mathworld.wolfram.com