

# Digraph Width Measures in Parameterized Algorithmics

Robert Ganian<sup>a</sup>, Petr Hliněný<sup>b</sup>, Joachim Kneis<sup>c</sup>, Alexander Langer<sup>c</sup>,  
Jan Obdržálek<sup>b</sup>, Peter Rossmanith<sup>c</sup>

<sup>a</sup>*Vienna University of Technology, Austria*

<sup>b</sup>*Faculty of Informatics, Masaryk University, Brno, Czech Republic*

<sup>c</sup>*Theoretical Computer Science, RWTH Aachen University, Germany*

---

## Abstract

In contrast to undirected width measures such as tree-width, which have provided many important algorithmic applications, analogous measures for digraphs such as directed tree-width or DAG-width do not seem so successful. Several recent papers have given some evidence on the negative side. We confirm and consolidate this overall picture by thoroughly and exhaustively studying the complexity of a range of directed problems with respect to various parameters, and by showing that they often remain NP-hard even on graph classes that are restricted very beyond having small DAG-width. On the positive side, it turns out that clique-width (of digraphs) performs much better on virtually all considered problems, from the parameterized complexity point of view.

*Keywords:* digraph, parameterized complexity, tree-width, DAG-width, DAG-depth, cycle rank, clique-width.

---

## 1. Introduction

The very successful concept of graph *tree-width* was introduced in the context of the Graph Minors project by Robertson and Seymour [RS86, RS91], and it turned out to be very useful for efficiently solving many graph problems (including NP-hard ones). In a nutshell, tree-width measures tree-likeness of a graph. Trees themselves, for example, have tree-width one and series-parallel graphs have tree-width two. Many graphs occurring in practical applications have small tree-width. This comes as no big surprise as one often deals with hierarchical structures that are inherently similar to trees. Examples include problems in VLSI design, evolution theory, interval routing, and the control-

---

*Email addresses:* rganian@gmail.com (Robert Ganian), hlineny@fi.muni.cz (Petr Hliněný), kneis@cs.rwth-aachen.de (Joachim Kneis), langer@cs.rwth-aachen.de (Alexander Langer), obdrzalek@fi.muni.cz (Jan Obdržálek), rossmani@cs.rwth-aachen.de (Peter Rossmanith)

flow graphs of structured programs. See [Bod93, Bod98, BK08, HOSG08] for surveys.

Tree-width is a property of *undirected graphs*. In this paper we will be interested in *digraphs* (directed graphs). Naturally, a width measure specifically tailored to digraphs with all the nice properties of tree-width would be tremendously useful. We can, of course, apply the concept of tree-width to digraphs, too, if we just forget the direction of all edges for the computation of the tree-width and the resulting *tree decomposition*. With such an approach we, however, seem to ignore too much: For example, in directed acyclic graphs (DAGs) it is easy to find a longest path while the problem is NP-complete in general. Nevertheless, DAGs have unbounded tree-width if we forget the directions.

On the search for a “truly directed” width measure inspired by tree-width, several suggestions were made, starting with directed tree-width [JRST01], and being complemented recently with several new approaches including directed path-width [Bar06], entanglement [BG05], D-width [Saf05], DAG-width [BDH<sup>+</sup>12] and Kelly-width [HK08].

Some positive results were encouraging: The Hamiltonian path problem can be solved in polynomial time if the directed tree width, the DAG-width, or the Kelly-width are bounded by a constant [JRST01]. More recently, it has been shown that parity games (Section 4.9) can be solved in polynomial time on digraphs of bounded entanglement [BG05], DAG-width [BDH<sup>+</sup>12] or Kelly-width [HK08].

Unfortunately, as encouraging as the first positive results are, there is also the negative side. For undirected graphs, the existence of a Hamiltonian path can be tested in *linear* time if the tree-width is bounded by a constant; only the constant hidden in the “big- $O$ ” increases with the tree-width. So this problem is *fixed-parameter tractable* (captured in the complexity class FPT, Section 2.3) for the parameter tree-width. While Hamiltonian path on digraphs is indeed solvable in polynomial time for bounded DAG-width, the degree of the polynomial in the running time increases with the DAG-width (in the complexity class XP, Section 2.3). This likely cannot be improved (unless the Exponential-time hypothesis fails) since Lampis, Kaouri, and Mitsou showed that Hamiltonian path is  $W[2]$ -hard for the parameter DAG-width [LKM11].

Even worse, many other natural problems remain NP-hard on digraphs of low widths [CD06, DGK09, KO11, LKM11] and some of them are already NP-complete on DAGs – such as MaxDiCut [LKM11] or oriented colouring [CD06]. This particularly implies that for DAG-width there cannot be a result similar to famous Courcelle’s MSO<sub>2</sub> theorem. Therefore, one should perhaps look for other new directed measures providing a “finer resolution” on DAGs.

We will add many more natural directed problems to the list, but will go even further: One of the main goals of this paper is to show that not only many problems are hard on DAGs, but rather that they remain hard even if we very severely further restrict the digraphs’ structure. To this end, we introduce *two new digraph measures*; K-width (Section 3.4) and DAG-depth (Section 3.3), with the intention to complete the full (and rather *negative*) picture of structural

digraph width parameters with some more restrictive ones.

On the other hand, one width measure that fares much better is *clique-width* [CO00], an algebraic width measure that equally handles graphs and digraphs (and related bi-rank-width generalizing the rank-width of undirected graphs [KR13]). Nearly all of our problems are fixed-parameter tractable or at least in XP with respect to this parameter. Even better, unlike as for DAG-width or Kelly-width, finding an optimal bi-rank-decomposition is known to be in FPT [HO08, KR13].

## 2. Preliminaries

### 2.1. Digraphs

We assume that the readers are familiar with standard terms of undirected graphs, for example in Diestel [Die05].

A *directed graph* (or *digraph*) is a pair  $(V, E)$  of disjoint sets of *vertices* and *arcs*, together with two mappings  $tail : E \rightarrow V$  and  $head : E \rightarrow V$  assigning to every arc  $e$  its *starting vertex*  $x = tail(e)$  and *terminal vertex*  $y = head(e)$  ( $e$  is said to be *directed* from  $x$  to  $y$ ). Note that a digraph may have several arcs between the same two vertices  $x, y$ . If two of them have the same direction (say from  $x$  to  $y$ ), they are called *parallel*. If  $x = y$ , then  $e$  is called a *loop*. We will sometimes refer an arc from  $x$  to  $y$  as to  $(x, y)$ .

A *directed path* is a digraph of the form:  $V = \{x_0, x_1, x_2, \dots, x_k\}$ ,  $E = \{(x_0, x_1), (x_1, x_2), \dots, (x_{k-1}, x_k)\}$  where  $x_i$  are all distinct; intuitively,  $x_0$  and  $x_k$  are called the endpoints of the directed path. The length of a directed path is defined as the number of its arcs. Two directed paths  $P_1$  and  $P_2$  are *internally disjoint* if they are vertex disjoint except for their endpoints. A *directed cycle* is a directed path with an additional arc  $(x_k, x_0)$ . A *directed acyclic graph (DAG)* is a digraph with no directed cycles.

Many terms of undirected graphs are naturally extended to digraphs, like those of subgraph/*subdigraph* and of *isomorphism*. Given two vertices  $x, y$ , we say that  $y$  is an *out-neighbour* (*in-neighbour*) of  $x$  if there exists an arc  $(x, y)$  ( $(y, x)$ , respectively). We say that  $y$  is reachable from  $x$  if there exists a directed path from  $x$  to  $y$ . A digraph  $G$  is *strongly connected* if each of its vertices is reachable from any other one. *Strong components* of  $G$  are the equivalence classes defined by the relation  $x \sim y$  meaning that  $x$  is reachable from  $y$  and  $y$  is reachable from  $x$ .

### 2.2. SAT and its variants

We define the Boolean Satisfiability problem (abbreviated as SAT). A *literal* is a positive propositional variable  $x$  or a negative variable  $\neg x$ . A *clause* is a finite set of literals, e.g.,  $C = x_1 \vee x_2 \vee \neg x_3$ . A propositional formula  $\phi$  in conjunctive normal form, or CNF formula for short, is a set of clauses  $\phi = C_1 \wedge \dots \wedge C_p$ . A CNF formula is a  $c$ -CNF formula if each clause contains at most  $c$  literals. Let  $var(\phi)$  denote the set of variables of  $\phi$ .

A CNF formula  $\phi$  is *satisfiable* if there is a truth assignment of  $\text{var}(\phi)$  for which  $\phi$  evaluates to true, otherwise  $F$  is *unsatisfiable*. SAT is the NP-complete problem of deciding whether a given CNF formula is satisfiable [Coo71, Lev73]. Analogously,  $c$ -SAT is the problem of deciding whether a given  $c$ -CNF formula is satisfiable. Based on the value of  $c$  we have the following distinction:

**Theorem 2.1** ([Kro67]). *2-SAT can be solved in polynomial time.*

**Theorem 2.2** ([GJ79], folklore). *The 3-SAT problem remains NP-complete even if the input CNF formula satisfies all the following conditions*

- every variable occurs in at most three clauses;
- no clause contains the same variable twice (not even positive and negative);
- no variable has all occurrences positive or all negative (and so it has at most two positive and at most two negative occurrences).

**Proof:** 3-SAT is one of Karp's basic NP-complete problems [GJ79]. If a variable  $x \in \text{var}(\phi)$  has  $k > 3$  occurrences in the clauses of  $\phi$  then, by a folklore trick, we replace these occurrences each with new variables  $x_i, i = 1, \dots, k$ , and add new clauses  $(x_1 \vee \neg x_2), (x_2 \vee \neg x_3), \dots, (x_k \vee \neg x_1)$  forcing these variables to have the same value in a satisfying assignment. If a variable  $x$  has all occurrences positive (all negative), then clauses containing  $x$  may be safely removed from the instance. Finally, if a clause contains  $x \vee \neg x$ , then again this clause may be safely removed. ■

### 2.3. Parameterized complexity

Parameterized complexity (see [DF99, FG06]) is an approach to describe the complexity of problems beyond the traditional methods as a function of the input size. To this end, the additional notion of a *parameter* derived from the input is used to provide a more fine-grained analysis of the time and space requirements to solve the problem on this input.

Formally, a parameterized problem  $P$  is a subset of  $\Sigma \times \mathbb{N}_0$ , where  $\Sigma$  is a finite alphabet. A parameterized problem  $P$  is said to be *fixed-parameter tractable* if there is an algorithm that given  $(x, k) \in \Sigma \times \mathbb{N}_0$  decides whether  $(x, k)$  is a yes-instance of  $P$  in time  $f(k) \cdot p(|x|)$  where  $f$  is some computable function of  $k$  alone,  $p$  is a polynomial and  $|x|$  is the size measure of the input. The class of such problems is denoted by FPT. The class XP is the class of parameterized problems that admit algorithms with a run-time of  $\mathcal{O}(|x|^{f(k)})$  for some computable  $f$ , that is polynomial-time for every fixed value of  $k$ .

Let  $P_k := \{(x, k) \mid (x, k) \in P\}$  be the  $k$ -slice of  $P$ . Note that if a problem  $P$  is in FPT or XP, then for each  $k \in \mathbb{N}$ , the  $k$ -slice  $P_k$  can be solved in polynomial time. For problems in FPT, the polynomial part of the running time is fixed, while for problems in XP the degree of the polynomial may depend on the parameter. In contrast, a problem  $P$  is *para-NP-hard* (para-NPH) or *para-NP-complete* (para-NPC) [FG06, p. 39] if there are finitely many integers  $k_1, \dots, k_l \in \mathbb{N}$  such that  $P_{k_1} \cup \dots \cup P_{k_l}$  is NP-hard or NP-complete, respectively.

Yet finer resolution is given by the levels of the so-called W-hierarchy  $\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \dots \subseteq \text{XP}$ , formally defined in terms of Boolean circuits with limited levels of unbounded fan-in (the *weft*) [DF99]. It is widely believed that the inclusions are proper; problems that are  $\text{W}[1]$ -hard under  $\text{fpt}$ -reductions are therefore presumably *fixed-parameter intractable* – meaning that XP-time algorithms are the “best possible” for them.

### 3. Digraph Width Measures

We first present a short recapitulation of existing width measures for digraphs, which is a necessary prerequisite for our main results in Section 4. We also introduce two new measures – DAG-depth and K-width, which (as noted above) are intended to complete the overall picture of directed structural width measures with some very restrictive ones.

#### 3.1. A brief list

Perhaps the oldest example of a digraph measure was given in the 1960’s by Eggan and Büchi [Egg63]:

**Definition 3.1 (Cycle rank).** The *cycle rank*  $cr(G)$  of a digraph  $G$  is defined inductively as follows: For DAGs,  $cr(G) = 0$ . If  $G$  is strongly connected and  $E(G) \neq \emptyset$ , then  $cr(G) = 1 + \min\{cr(G - v) : v \in V(G)\}$ . Otherwise,  $cr(G)$  is the maximum over the cycle ranks of the strongly connected components of  $G$ .

We remark that one can, from this definition, compute  $cr(G)$  by an XP algorithm when parameterized by  $cr(G)$ .

A related measure can straightforwardly be derived from the long-time known *feedback vertex set* problem as suggested, e.g., in [Nie10] for undirected graphs.

**Definition 3.2 (DFVS-number).** The *directed feedback vertex set number*  $dfn(G)$  of a digraph  $G$  is the minimum cardinality of a set  $S \subseteq V(G)$  such that  $G - S$  is a DAG.

**Proposition 3.3.** *For every digraph  $G$  it holds  $cr(G) \leq dfn(G)$ .*

**Proof:** This follows from Definition 3.1 by simple induction on  $dfn(G)$ . ■

Another example of a width measures (with an algebraic / logic flavor) is that of clique-width by Courcelle and Olariu [CO00], better known over undirected graphs but originally defined over digraphs as well as graphs:

**Definition 3.4 (Clique-width).** A *k-expression* is an algebraic expression with the following four operations on vertex-labeled (di)graphs using  $k$  labels: create a new vertex with label  $i$ ; take the disjoint union of two labeled graphs; add all edges/arcs between vertices of label  $i$  and label  $j$ ; and relabel all vertices with label  $i$  to have label  $j$ . The *clique-width* of a (di)graph  $G$  equals the minimum  $k$  such that (some labeling of)  $G$  is the value of a  $k$ -expression.

While there is no known exact parameterized algorithm for computing the clique-width of a digraph, it can be approximated by so called *bi-rank-width* of Kanté and Rao [KR13]. Bi-rank-width is computable, parameterized by itself, by an FPT algorithm [HO08, KR13]. We, however, use clique-width in this paper due to its easier definition.

Aside of the aforementioned three measures, a stream of new “structural” digraph width measures occurred in the past decade, influenced by the great success of tree-width in undirected graphs. This started with directed tree-width [JRST01], and has been then complemented with several other approaches including entanglement [BG05], D-width [Saf05], directed path-width [Bar06], DAG-width [BDH<sup>+</sup>12], and Kelly-width [HK08]. Due to restricted space in this paper, we skip the mostly lengthy definitions which can be found in the cited papers. For the purpose of our paper, it appears most useful to formulate alternative game-theoretic characterizations of some of the measures in the next section.

### 3.2. Cops-and-robber games on digraphs

The cops-and-robber game was defined by Seymour and Thomas [ST93] as an useful alternative characterization of tree-width. In the directed version (as well as the undirected) version of this game, there are  $k$  cops (controlled by one player) and a robber (controlled by the other). Each cop can either move around in a helicopter or land on and occupy a vertex of the graph, and the robber occupies a vertex. The robber can, however, see the helicopter landing, and can move at a great speed along a cop-free directed path to another vertex. The objective of the cops is to capture the robber by landing on the vertex currently occupied by him, the objective of the robber is to avoid capture. (The point of the helicopters is that, unlike the robber, cops are not restricted to move along the arcs of the graph.) Formally, the game is defined below, where  $[X]^{\leq k}$  is the set of all subsets of  $X$  of size at most  $k$ .

**Definition 3.5 (Cops and robber game).** Given a digraph  $G := (V, E)$ , the  $k$ -cops-and-robber game on  $G$  is played between two players: the *cop* and the *robber*. A position of this game is a pair  $(X, r)$ , where  $X \in [V]^{\leq k}$  are the vertices occupied by the cops and  $r \in V$  is the vertex occupied by the robber. The game is played in turns, as follows:

- At the beginning, the cop player chooses  $X_0 \in [V]^{\leq k}$ , and the robber player chooses a vertex  $r_0 \in V$ , giving position  $(X_0, r_0)$ .
- From position  $(X_i, r_i)$ , if  $r_i \notin X_i$  then the cop player chooses  $X_{i+1} \in [V]^{\leq k}$ , and the robber player chooses a vertex  $r_{i+1}$  such that there is a directed path from  $r_i$  to  $r_{i+1}$  in the digraph  $G \setminus (X_i \cap X_{i+1})$ .
- A *play* in the game is a maximal (finite or infinite) sequence  $\pi := (X_0, r_0), (X_1, r_1), \dots$  of positions given by the rules above.

- A play  $\pi$  is *winning for the cop player* if it is finite. (Note that, by the rules above, this implies that  $r_m \in X_m$  for the last position  $(X_m, r_m)$  of this play.) A play  $\pi$  is *winning for the robber player* if it is infinite.
- A (*k-cop*) *strategy* for the cop player is a function  $f$  from  $[V]^{\leq k} \times V$  to  $[V]^{\leq k}$ . A play  $(X_0, r_0), (X_1, r_1), \dots$  is *consistent* with a strategy  $f$  if  $X_{i+1} = f(X_i, r_i)$  for all  $i$ . The strategy  $f$  is called a *winning strategy* if every play consistent with  $f$  is winning for the cop player.

Variants of the game where the robber moves first, or only one cop can be moved at a time, or the cops are lifted and placed in separate moves are all easily seen to be equivalent in that the cop number of a digraph does not depend on the variant.

In addition to 'vanilla' cops-and-robber games we can consider variants which change the descriptive power of these games. One alternative is to restrict the cop strategy in such a way, that a cop is never placed on a vertex from which a cop was previously removed. Such strategies are called to be (*cop*) *monotone*. Another possibility is to make the robber *invisible* to cops. In that case they have to exhaustively search the graph, and their strategy has to depend on the history of the play so far. These and other alternatives mentioned below allow us to capture many different digraph with measures.

We begin by showing a cops-and-robber game characterization of cycle rank. We say that a cop strategy is *lift-free* if the cop player never moves a cop from a vertex once he has landed. Formally,  $X_i \subseteq f(X_i, r_i)$  for all  $i$  (note that this trivially implies monotonicity).

**Proposition 3.6.** *For any digraph  $G$ , the cycle rank of  $G$  is at most  $k$  if, and only if, the cop player has a lift-free winning strategy in the  $(k + 1)$ -cops-and-robber game on  $G$  against a visible robber who is bound to stay in the same strong component of the cop-free induced subgraph.*

**Proof:** We proceed by induction on  $|V(G)|$  along the definition of  $cr(G)$ . If  $G$  is a DAG ( $cr(G) = 0$ ), then the robber cannot move at all, and so 1 cop simply lands on him and the game is over. Suppose that  $G$  is strongly connected. Then, for some  $w \in V(G)$ , it is  $cr(G) = 1 + cr(G - w)$  by the definition. By inductive assumption, the cop player has a winning strategy with  $cr(G - w) + 1$  cops on  $G - w$ , and so he wins—landing first on  $w$ , on  $G$  with  $1 + cr(G - w) + 1 = cr(G) + 1$  cops. On the other hand, for any first move  $u \in V(G)$  of the cop player, the robber may position himself (before actual cop landing) and play his optimal strategy on  $G - u$  by induction, showing that  $1 + cr(G - u) + 1 \geq 1 + \min\{cr(G - v) : v \in V(G)\} + 1 = cr(G) + 1$  cops are needed.

Suppose that  $G$  has more than one strong components, denoted by  $G_1, \dots, G_a$ . The robber has to stay within one of them, say  $G_i$ . By induction, the cop player has a winning strategy with  $1 + cr(G_i) \leq 1 + \max_{j=1, \dots, a} cr(G_j) = cr(G) + 1$  cops. Conversely, robber's strategy is to initially choose any vertex of  $G_j$  maximizing  $cr(G_j)$ , showing that  $cr(G) + 1$  cops are indeed necessary. ■

We continue with some of the previously mentioned measures. For *directed tree-width* ( $dtw$ ) [JRST01], in the related cops-and-robber game the robber is again bound to stay within the same strong component of the cop-free induced subgraph, but the cop strategy need not be lift-free. However, the relationship [JRST01] between the number of cops and the directed tree-width is not tight.

By [Bar06], *directed path-width* ( $dpw$ ) of a digraph  $G$  is at most  $k$  if, and only if, the cop player has a monotone winning strategy on  $G$  in the  $(k + 1)$ -cops-and-robber game against an (unrestricted) invisible robber. Similarly, in [HK08] it is proved that the *Kelly-width* ( $kellyw$ ) of a digraph  $G$  is at most  $k$  if, and only if, the cop player has a monotone winning strategy on  $G$  in the  $k$ -cops-and-robber game against an invisible “lazy” robber which is allowed to move only if a cop is about to land on his position. No analogous characterizations of entanglement ( $ent$ ) and D-width have been published so far.

In the perspective of our paper the prime position is given to DAG-width:

**Theorem 3.7 (DAG-width [BDH<sup>+</sup>12]).** *For any digraph  $G$ , the DAG-width of  $G$  is at most  $k$  if, and only if, the cop player has a monotone winning strategy in the  $k$ -cops-and-robber game on  $G$  (against a visible unrestricted robber).*

We compare these measures in the following brief summary.

**Theorem 3.8.** *Let  $G$  be a digraph. Then the following inequalities hold:*

$$\begin{aligned} 1/3 (dtw(G) - 1) &\leq_{[BDH^+12]} dagw(G) \leq dpw(G) + 1 \leq \\ &\leq_{[Gru08]} cr(G) + 1 \leq dfn(G) + 1 \\ 1/6 (dtw(G) + 2) &\leq_{[HK08]} kellyw(G) \leq dpw(G) + 1 \\ 1/3 (dtw(G) + 2) &\leq_{[JRST01]} ent(G) \leq_{[Rab08]} dpw(G) \end{aligned}$$

Moreover, when DAG-width is bounded, so is Kelly-width [HO06].

Notice, in particular, that if a problem is hard for graphs of bounded cycle rank, then it is hard for all the other measures in Theorem 3.8, except possibly the DFVS-number which is even more restrictive. Conversely, a problem is solvable for graphs which have any of these measures bounded in at most the same complexity class as for graphs of bounded directed tree-width. In view of this fact, and taking into an account the limited algorithmic usability of entanglement and the many technical difficulties surrounding directed tree-width [Adl07], we have chosen DAG-width, cycle rank, and DFVS-number as the representatives of all the aforementioned measures in our survey.

In addition to that, we will consider also clique-width (Definition 3.4) which has nearly no relation to the other measures, and two newly introduced measures DAG-depth (Definition 3.9) and K-width (Definition 3.13) aiming at finer resolution even on DAGs.

### 3.3. DAG-depth

This new concept has been inspired by the tree-depth notion of Nešetřil and Ossona de Mendez. In their paper, [NdM06, Lemma 2.2] gives an alternative



inductive definition of the *tree-depth*  $td(G)$  of undirected  $G$  as follows (compare to Def. 3.1). If  $|V(G)| = 1$ , then  $td(G) = 1$ . If  $G$  is connected, then  $td(G) = 1 + \min\{td(G-v) : v \in V(G)\}$ . Otherwise,  $td(G)$  equals the maximum over the tree-depth of the connected components of  $G$ . This definition easily corresponds to a lift-free cop search strategy in the cops-and-robber game on an undirected graph (see also another later game characterization in [GT11]).

We propose a new “directed” generalization of this definition. For a digraph  $G$  and any  $v \in V(G)$ , let  $G(v)$  denote the subdigraph of  $G$  induced by the vertices reachable from  $v$ . We call *reachable fragments* of  $G$  the maximal elements of the poset  $\{G(v) : v \in V(G)\}$  which is ordered by inclusion (“being subgraph of”). Notice that reachable fragments in the undirected case coincide with connected components.

**Definition 3.9 (DAG-depth).** The *DAG-depth*  $ddp(G)$  of a digraph  $G$  is inductively defined as follows: If  $|V(G)| = 1$ , then  $ddp(G) = 1$ . If  $G$  has a single reachable fragment, then  $ddp(G) = 1 + \min\{ddp(G-v) : v \in V(G)\}$ . Otherwise,  $ddp(G)$  equals the maximum over the DAG-depth of the reachable fragments of  $G$ .

**Proposition 3.10.** *If  $G$  is a symmetric digraph, then  $ddp(G) = cr(G) + 1$ . On the other hand, there exist DAGs with arbitrarily high DAG-depth.*

**Proof:** The first part simply follows by comparing Definitions 3.1 and 3.9, while the second one is witnessed by a long directed path (see below). ■

The key to understanding and using DAG-depth, again, lies in a natural game characterization:

**Theorem 3.11.** *The DAG-depth of a digraph  $G$  is at most  $k$  if and only if the cop player has a lift-free winning strategy in the  $k$ -cops-and-robber game on  $G$ .*

**Proof:** We proceed by induction on  $|V(G)|$  along the definition of DAG-depth. That is trivial if  $|V(G)| = 1$ . Otherwise, let  $R_1, \dots, R_d$  be all the reachable fragments of  $G$ . We first consider  $d > 1$ , and recall  $ddp(G) = \max\{ddp(R_i) : i = 1, \dots, d\} = ddp(R_j)$ . Since the robber may start in a vertex reaching whole  $R_j$ , the cop player cannot have a winning strategy with  $k < ddp(R_j)$  by inductive assumption. Conversely, a winning strategy for the cop player with  $k = ddp(G)$  is easy; if the robber is in a vertex of some  $R_i$  (and cannot reach  $G - V(R_j)$  by the definition), the cop player uses his strategy for  $R_i$ .

Second, assume  $d = 1$  and  $ddp(G) = 1 + \min\{ddp(G-v) : v \in V(G)\} = 1 + ddp(G-w)$ . In particular, there is  $u \in V(G)$  reaching all the vertices of  $G$ . Robber’s strategy is to start in  $u$  and move according to his strategy in  $G - v$  whenever a cop is about to land on  $v \in V(G)$ , showing  $k \geq 1 + ddp(G-v)$ . Conversely, a winning strategy for the cop player starts by landing on  $w$  and thus uses  $k = 1 + ddp(G-w) = ddp(G)$  cops by inductive assumption. ■

**Corollary 3.12.** *a) For any subdigraph  $G'$  of  $G$ , it is  $ddp(G') \leq ddp(G)$ .*

- b) For any digraph  $G$ , the DAG-depth of  $G$  is greater than or equal to the DAG-width and the cycle rank plus one of  $G$ .
- c) If  $P$  is a directed path, then  $ddp(P) = \lfloor \log_2 |V(P)| \rfloor + 1$ .
- d) If  $\ell$  is the number of vertices of a longest directed path in a digraph  $G$ , then  $\lfloor \log_2 \ell \rfloor + 1 \leq ddp(G) \leq \ell$ .
- e) The DAG-depth of a given digraph  $G$  can be approximated within exponential margins by an FPT algorithm, and computed exactly by an XP algorithm.

**Proof:** a) The cop player may use the restriction of his  $G$ -strategy to  $G'$ .

b) One simply compares the games from Theorems 3.7, 3.11 and Proposition 3.6: A lift-free strategy is monotone by the definition.

c) Let  $d(\ell) = ddp(P)$  where  $\ell = |V(P)|$ , and  $d(0) = 0$ . Note that  $d$  is a non-decreasing integer function by a). A directed path always consists of a single reachable fragment, and  $P - v$  consists of one or two paths on, say,  $i$  and  $\ell - 1 - i$  vertices. Therefore, by Definition 3.9, it is  $d(1) = 1$  and  $d(\ell) = 1 + \min_{i=0 \dots \ell-1} \max\{d(i), d(\ell - 1 - i)\} = 1 + d(\lceil (\ell - 1)/2 \rceil)$ . A routine solution of this recursion gives  $d(\ell) = \lfloor \log_2 \ell \rfloor + 1$ .

d) The lower bound follows from a) and c). We describe a simple  $\ell$ -move lift-free winning strategy for the cop player on any such digraph  $G$ . The first cop lands on the initial position  $s_1$  of the robber. In cop move  $i > 1$ , the cop number  $i$  lands on a vertex  $s_i$  of  $G$  which is the out-neighbour of  $s_{i-1}$  on some directed path from  $s_{i-1}$  to the current robber position. Since all directed paths starting in  $s_1$  have  $\leq \ell$  vertices, the robber is finally caught after  $\leq \ell$  cop moves.

e) We first compute the longest directed path length  $\ell$  in  $G$ , which can be done by an FPT algorithm, e.g., [CKL<sup>+</sup>09]. This  $\ell$  is already a sufficient estimate of the DAG-depth of  $G$  by d). In the second part, we carry out a brute-force recursive computation of the DAG-depth of  $G$  according to Definition 3.9 (the depth of recursion is bounded by  $\ell$ ). ■

### 3.4. $K$ -width

Another attempt to define a “finer” digraph width measure is the following:

**Definition 3.13 (K-width).** The  $K$ -width (a shortcut of “Kenny width”) of a digraph  $G$  is the maximum number of distinct (not necessarily disjoint) directed  $s$ - $t$  paths in  $G$  over all pairs of distinct vertices  $s, t \in V(G)$ .

Similarly to DAG-depth in Proposition 3.10,  $K$ -width can be arbitrarily large on DAGs, and despite its seemingly different nature,  $K$ -width is also related to previous game characterizations.  $K$ -width is, though, generally incomparable with cycle rank and directed path-width which are both unbounded on symmetric orientations of trees.

**Theorem 3.14.** For any digraph  $G$ , the  $K$ -width of  $G$  is greater or equal to the DAG-width of  $G$  minus one.

**Proof:** We are going to use Theorem 3.7. Let  $T$  be any depth first search tree of  $G$ . Based on  $T$ , we outline a monotone search strategy for the cop player on  $G$ , in which the player is to use a cop number  $k + 1$  only if there are at least  $k$  paths between a pair of vertices.

- (i) In the first move a cop is placed at the root of  $T$ .
- (ii) In each subsequent cop-placing move, the cop player chooses the (unique) vertex  $v$  of  $G$  such that  $v$  is an out-neighbour of a cop-occupied vertex, and  $v$  reaches the robber along a cop-free path in  $T$ .
- (iii) Whenever a cop-occupied vertex  $u$  is no longer reachable from the current robber position, the cop from  $u$  is lifted back.

This strategy is clearly monotone. Consider the vertex  $v$  in rule (ii). If there was a cop-occupied vertex  $w$  in  $G$  which is not an ancestor of  $v$ , then  $w$  must no longer be reachable by the robber since  $T$  is a DFS tree. So (iii) for  $u = w$  applies before (ii). Therefore, our strategy maintains an invariant that all vertices occupied by cops belong to one directed path of  $T$ .

Consider now a situation when there is a set  $U$  of  $k$  cop-occupied vertices in  $G$ , and rule (ii) applies again. Then there is a path  $P \subseteq T$  such that  $U \subseteq V(P)$ . Let  $s$  be the last cop-occupied vertex of  $P$ . By (iii), each vertex  $w \in U$  is reachable in  $G$  from the robber vertex  $r$  along a cop-free path  $Q_w$ . So  $P \cup Q_w$  contains a path from  $r$  to  $s$  and these  $k$  paths are pairwise distinct for distinct  $w$ .

■

**Proposition 3.15.** *Let  $G$  be a digraph and  $t$  equals the  $K$ -width of  $G$ .*

- a) *For any  $u \in V(G)$ , all (at most  $t \cdot |V(G)|$ ) directed paths starting at  $u$  can be enumerated in time  $O(t \cdot |E(G)|)$  without prior knowledge of  $t$ .*
- b) *The value  $t$  of  $K$ -width can be computed in time  $O(t \cdot |V(G)| \cdot |E(G)|)$ .*

**Proof:** a) We enumerate the paths in  $G$  starting at  $u$  by backtracking, and prune the search whenever finding a vertex that is already on the current path. The resulting search tree has at most  $t \cdot |V(G)|$  nodes: Each node in the search tree corresponds to a simple path in  $G$  starting at  $u$ . There can be at most  $t$  such paths with the same terminal vertex. The time spent in each node of the search tree is  $O(d)$ , where  $d$  is the out-degree of the terminal vertex of the corresponding simple path. Overall this amounts to a running time of  $O(t \cdot |E(G)|)$ .

b) For every vertex  $u \in V(G)$ , we call the algorithm of a) and compute the numbers of paths from  $u$  to each of the vertices in  $V(G) \setminus \{u\}$ . The maximum number we encounter over all choices of  $u$  is exactly  $t$ . ■

### 3.5. Simple comparison and differences

Finally, we couple the many relations between considered measures claimed in 3.8, 3.12, 3.14 with a simple series of examples demonstrating differences between the measures in Table 1.

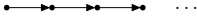




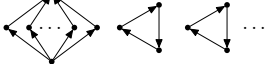

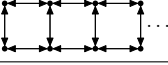
Graph family	DAG-depth	K-width	DFVS-number	cycle rank	DAG-width
	$\infty$	1	0	0	1
	3	$\infty$	0	0	1
	$\infty$	$\infty$	0	0	1
	3	1	$\infty$	1	2
	$\infty$	1	$\infty$	1	2
	3	$\infty$	$\infty$	1	2
	$\infty$	1	$\infty$	$\infty$	3
	$\infty$	$\infty$	$\infty$	$\infty$	3

Table 1: Families of graphs demonstrating various possible combinations of the listed width measures being bounded and unbounded.

## 4. Common Problems Parameterized by Digraph Measures

### 4.1. $MSO_1$ model checking ( $\phi$ - $MSO_1MC$ )

Before dealing with specific problems on digraphs, we first introduce a useful general problem framework based on  $MSO$  (monadic second order) logic on digraphs. Specifically, we consider the variant abbreviated as  $MSO_1$ , which uses propositional logic, variables for digraph vertices and vertex sets, predicate  $arc(u, v)$  for arcs of the digraph, and quantification over vertices and their sets. This is best illustrated with the following examples:

**Example 4.1.** *The following properties are expressible in  $MSO_1$  on digraphs:*

- a directed dominating set  $X$  as  $\forall z(z \in X \vee \exists x \in X arc(x, z))$ ,
- the existence of a kernel  $S$  as  $\exists S \forall x[x \notin S \leftrightarrow (\exists y \in S arc(x, y))]$ , and
- a feedback vertex set  $Z$  as  $\forall X[X \cap Z = \emptyset \rightarrow (\exists x \in X \forall y \in X \neg arc(x, y))]$ .

On the other hand,  $MSO_1$  cannot express existence of Hamiltonian cycle, for instance (cf. page 113 of [EF99]).

The  $MSO_1$  model checking problem ( $\phi$ - $MSO_1MC$ ) inputs a digraph  $G$ , and the task is to decide whether  $G \models \phi$  where the  $MSO_1$  sentence  $\phi$  is a fixed part of the problem definition.

We also briefly introduce the more general *LinEMSO<sub>1</sub> optimization problems* as given in [CMR00]. Consider any MSO<sub>1</sub> formula  $\psi(X_1, \dots, X_p)$  with free set variables, and state the following problem on an input (di)graph  $G$ :

$$\text{opt}\{f_{lin}(W_1, \dots, W_p) : W_1, \dots, W_p \subseteq V(G), G \models \psi(W_1, \dots, W_p)\},$$

where *opt* can be min or max, and  $f_{lin}$  is a linear evaluational function. It is

$$f_{lin}(W_1, \dots, W_p) = \sum_{i=1}^p \sum_{j=1}^m \left( a_{i,j} \cdot \sum_{x \in W_i} f_j(x) \right) \quad (1)$$

where  $m$  and  $a_{i,j}$  are (integer) constants and  $f_j$  are (integer) weight functions on the vertices of  $G$ . Typically  $f_{lin}$  is just the cardinality function. Such as,

$$\psi(X) \equiv \forall v, w (v \notin X \vee w \notin X \vee \neg \text{edge}(v, w)) \quad \text{and} \quad \text{“max } |X| \text{”}$$

describes the maximum independent set problem, or

$$\psi(X) \equiv \forall z (z \in X \vee \exists x \in X \text{ arc}(x, z)) \quad \text{and} \quad \text{“min } |X| \text{”}$$

is the minimum directed dominating set problem.

In full generality one gets the following:

**Theorem 4.2 ([CMR00] for the undirected case).** *For every integer  $t$  and MSO<sub>1</sub> formula  $\psi$ , every  $\psi$ -LinEMSO<sub>1</sub> optimization problem is fixed-parameter tractable on digraphs of clique-width  $t$ , with the parameters  $t$  and  $|\psi|$ .*

The current standing of Theorem 4.2 is split; since, on the one hand, the original proof of the undirected version by Courcelle, Makowsky, and Rotics [CMR00] has been followed by at least two different published proofs in [GH10, KLR11], while on the other hand, none of these published proofs explicitly includes the directed case. Yet, a formal proof of the directed case is a simple translation of any one of these previously published arguments into digraph terms, and so it does not really constitute a new result. For the sake of completeness we include a short alternative proof of Theorem 4.2 via a reduction of the directed version into the undirected one with vertex labels [CMR00].

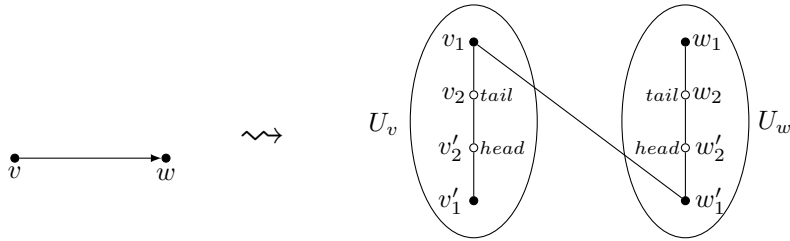


Figure 1: Reducing the directed version of Theorem 4.2 into an undirected one.

**Proof:** For an input digraph  $G$  of clique-width  $t$ , we construct an undirected labeled graph  $H$  of clique-width  $\leq 4t$ , with labels *head*, *tail* given to some of its

vertices: For each  $v \in V(G)$  we create a new 4-set of vertices  $U_v = \{v_1, v_2, v'_2, v'_1\}$  inducing a path of length three with the edges  $v_1v_2, v_2v'_2, v'_2v'_1$ ; where  $v_2$  is given the label *tail* and  $v'_2$  *head*. See Fig. 1. From the disjoint union of  $U_v$  for  $v \in V(G)$ , we form  $H$  by adding the edges  $v_1w'_1$  whenever  $(v, w) \in E(G)$ .

We accordingly translate a directed  $\text{MSO}_1$  formula  $\psi$  into an undirected one  $\sigma$  as follows: Every vertex variable  $x$  in  $\psi$  is replaced with a set variable  $Z_x$  in  $\sigma$  (intended to describe the aforementioned set  $U_x$  of  $H$ ), specified by  $\text{single}(Z_x) \equiv \lvert Z_x \rvert = 4 \wedge \exists x_1, x_2, x'_2, x'_1 \in Z_x [ \text{tail}(x_2) \wedge \text{head}(x'_2) \wedge \text{edge}(x_2, x'_2) \wedge \text{edge}(x_1, x_2) \wedge \text{edge}(x'_2, x'_1) ]$ . Literally,  $\exists x[\dots]$  is translated into  $\exists Z_x [ \text{single}(Z_x) \wedge \dots ]$ . Then  $x \in X$  in  $\sigma$  is simply replaced with  $Z_x \subseteq X$  (while set variables are literally untouched), and  $\text{arc}(x, y)$  is replaced with  $t\text{-arc}(Z_x, Z_y) \equiv \exists x_1, x_2 \in Z_x \exists y'_1, y'_2 \in Z_y [ \text{tail}(x_2) \wedge \text{head}(y'_2) \wedge \text{edge}(x_1, y'_1) \wedge \text{edge}(x_1, x_2) \wedge \text{edge}(y'_1, y'_2) ]$ . In a result,  $G \models \psi \iff H \models \sigma$ .

Regarding the  $\psi$ -LinEMSO<sub>1</sub> optimization framework (1), we set  $f'_j(v_1) := f_j(x)$  and  $f'_j(v_2) = f'_j(v'_2) = f'_j(v'_1) = 0$  for each  $U_v = \{v_1, v_2, v'_2, v'_1\}$  from the construction of  $H$ . We moreover literally translate each set quantifier  $\exists X[\dots]$  of  $\psi$  into  $\exists X [ \forall x \in X \exists Z \subseteq X ( \text{single}(Z) \wedge x \in Z ) \wedge \dots ]$  (a ‘‘sanity’’ condition) for  $\sigma$ . Then every solution of  $G \models \psi(W_1, \dots, W_p)$  is in a one-to-one correspondence to the solution  $H \models \sigma(W'_1, \dots, W'_p)$  such that  $W'_i = \bigcup_{w \in W_i} U_w$  for  $i = 1, \dots, p$  and  $f_{lin}(W_1, \dots, W_p) = f'_{lin}(W'_1, \dots, W'_p)$ .

We now solve the derived  $\sigma$ -LinEMSO<sub>1</sub> optimization problem on undirected  $H$  using the algorithms of [CMR00, HO08] since the size of  $H$  is linear in that of  $G$  and  $H$  is of bounded clique-width as well.  $\blacksquare$

Theorem 4.2 particularly implies that the problems listed in Example 4.1 (and many others) are in FPT on digraphs parameterized by clique-width. No analogous result, however, seem possible for our other width measures in general:

**Proposition 4.3.** *There exists an  $\text{MSO}_1$  sentence  $\phi$  such that the  $\phi$ - $\text{MSO}_1\text{MC}$  problem is NP-hard even on DAGs that are of  $K$ -width 1 and DAG-depth 2.*

**Proof:** We prove this by a reduction from the 3-colourability problem of undirected graphs [GJ79] (however, many other NP-hard problems can be used here as well). Given a graph  $H$  without isolated vertices, we construct an acyclic digraph  $G$  of  $K$ -width 1 and DAG-depth 2: For every edge  $e = uv$  of  $H$ , we add a new vertex  $x_e$  and replace  $e$  with two arcs  $ux_e, vx_e$ . Then we simply write  $\text{edge}(u, v) \equiv \exists x(\text{arc}(u, x) \wedge \text{arc}(v, x))$  to ‘‘interpret’’ the edges of  $H$  in  $G$ . 3-colourability of  $H$  is then expressed in  $G$  as follows:  $\exists X_1, X_2, X_3 [ \forall u (\bigvee_{i=1,2,3} u \in X_i) \wedge \forall u, v (\text{edge}(u, v) \rightarrow \neg \bigvee_{i=1,2,3} (u, v \in X_i)) ]$ .  $\blacksquare$

#### 4.2. Hamiltonian Path (HAM) and Longest Path

Many problems on digraphs are concerned with finding paths with certain properties. For start, the classical NP-hard *Hamiltonian Path* (HAM) problem is to find a directed path that visits each vertex of a digraph exactly once. A natural generalization of HAM is the *Longest Path* problem (LONGEST PATH),

where one is asked to find the longest directed path in a given digraph. Start and end vertices are not specified in these problems.

**Theorem 4.4.** *The HAM problem on digraphs*

- a) *has an XP algorithm parameterized by directed tree-width and, consequently, by DAG-width (Johnson, Robertson, Seymour, and Thomas [JRST01])—this assumes the corresponding decomposition is given along with the input;*
- b) *has an XP algorithm parameterized by clique-width ([GHO13]);*
- c) *is W[2]-hard when parameterized by DAG-width and cycle rank (Lampis, Kaouri, and Mitsou [LKM11]), and W[1]-hard when parameterized by clique-width (Fomin, Golovach, Lokshantov, and Saurabh [FGLS10b]).*

We prove our simple new FPT results for the parameters K-width and DAG-depth on more general LONGEST PATH:

**Theorem 4.5.** *There is a fixed-parameter tractable algorithm solving the LONGEST PATH and HAM problems on a digraph  $G$*

- a) *in time  $O(t \cdot |V(G)| \cdot |E(G)|)$  if  $G$  is of K-width at most  $t$ ;*
- b) *in time  $O(4^{2^t + O(t^3)} \cdot |V(G)| \cdot |E(G)|)$  if  $G$  is of DAG-depth at most  $t$ .*

*This holds also if  $t$  is unknown to the algorithm.*

**Proof:** a) For all  $u \in V(G)$  we enumerate all directed paths starting at  $u$  according to Proposition 3.15 while keeping track of their lengths.

b) We know by Corollary 3.12 that  $\lfloor \log_2 \ell \rfloor + 1 \leq t$ , or in other words,  $\ell \leq 2^t - 1$ , where  $\ell$  is the (unknown) number of vertices of the longest directed path. We can hence use an arbitrary FPT-algorithm for the Longest Path decision problem in the standard parameterization (e.g., [CKL<sup>+</sup>09] with running time  $4^{\ell + O(\log^3 \ell)} |V(G)| \cdot |E(G)|$ ): We begin with  $\ell = 1$  and subsequently increase  $\ell$  until a “no”-instance is found.

In the HAM problem we ask for a directed path of length  $|V(G)| - 1$ . ■

#### 4.3. Vertex-disjoint Paths Problem ( $k$ -PATH, $c$ -PATH)

Another problem is *Vertex-disjoint Paths* ( $k$ -PATH); given a digraph and  $k$  pairs of nodes  $(s_i, t_i)$ ,  $1 \leq i \leq k$ , the task is to find pairwise disjoint directed paths from each  $s_i$  to the respective  $t_i$ . To distinguish between bounded and unbounded values of  $k$ , we denote by  $c$ -PATH the variant of the Vertex-disjoint Paths problem in which  $c = k$  is a problem constant (rather than a part of the input).

**Theorem 4.6.** *The  $k$ -PATH problem (with  $k$  as part of input)*

- a) *is NP-complete on DAGs (implicit in the reduction of Even, Itai, and Shamir [EIS76]);*

- b) when parameterized by  $k$  on DAGs, it has an XP algorithm (Fortune, Hopcroft, and Wyllie [FHW80]), but remains  $W[1]$ -hard (Slivkins [Sli10]);
- c) is NP-complete on undirected graphs of clique-width 6 (Gurski and Wanke [GW06]), and hence consequently on digraphs of clique-width 6.

The  $c$ -PATH problem;

- d) is NP-complete for any  $c \geq 2$  (Fortune, Hopcroft, and Wyllie [FHW80]);
- e) has an XP algorithm with respect to the directed tree-width and DAG-width ([JRST01])—this assumes the decomposition given along with the input.

We, furthermore, informally remark that there exists a “mixed” generalization of  $c$ -PATH which remains NP-complete [BJK09] on DAGs.

**Proposition 4.7.** *Let  $G$  be a digraph, and let  $c$  pairs of vertices  $s_i, t_i \in V(G)$ ,  $i = 1, \dots, c$  be given. There is an  $MSO_1$  formula (depending on  $c$ ) expressing the existence of  $c$  pairwise disjoint directed  $s_i$ - $t_i$  paths,  $i = 1, \dots, c$ , in  $G$ . Hence the  $c$ -PATH problem is in FPT when parameterized by clique-width.*

**Proof:** We write

$$\exists X_1, \dots, X_c \left[ \bigwedge_{i \neq j \in \{1, \dots, c\}} X_i \cap X_j = \emptyset \wedge \bigwedge_{i \in \{1, \dots, c\}} s_i, t_i \in X_i \wedge \bigwedge_{i \in \{1, \dots, c\}} \forall Z \subseteq X_i \left( (s_i \in Z \wedge t_i \notin Z) \rightarrow \exists x \in Z, y \in X_i \setminus Z (\text{arc}(x, y)) \right) \right]$$

which means that there exist pairwise disjoint sets  $X_1, \dots, X_c \subseteq V(G)$  such that  $s_i, t_i \in X_i$ , and each  $X_i$  induces a subdigraph of  $G$  in which  $t_i$  is reachable from  $s_i$ . Notice that  $s_i, t_i$  are not variables, but constants in this sentence. The rest follows from Theorem 4.2. ■

Our two new width parameters are of limited (yet nontrivial) help; as we summarize in Theorems 4.8 and 4.9.

**Theorem 4.8.** *The  $k$ -PATH problem (with  $k$  as part of input)*

- a) can be solved in polynomial time on digraphs of  $K$ -width or DAG-depth 2;
- b) is NP-complete on DAGs of  $K$ -width 3 and DAG-depth 4.

**Proof:** a) Given a digraph  $G$  with  $K$ -width  $\leq 2$  and  $k$  pairs of nodes  $(s_1, t_1), \dots, (s_k, t_k)$ , we first for every  $1 \leq i \leq k$  compute by Proposition 3.15 the two possible paths  $p_{i,1}$  and  $p_{i,2}$  from  $s_i$  to  $t_i$ . Then we construct a 2-SAT formula as follows: For each pair  $(s_i, t_i)$ ,  $1 \leq i \leq k$ , there is a clause over the two alternative paths,  $C_i = (p_{i,1} \vee p_{i,2})$ . Furthermore, for each pair of non-disjoint paths  $p_1, p_2 \in \{p_{i,j} : 1 \leq i \leq k, 1 \leq j \leq 2\}$ , such that  $p_1 \neq p_2$  and  $V(p_1) \cap V(p_2) \neq \emptyset$ , there is a clause excluding each other,  $C_{p_1, p_2} = (\neg p_1 \vee \neg p_2)$ . Then we solve the



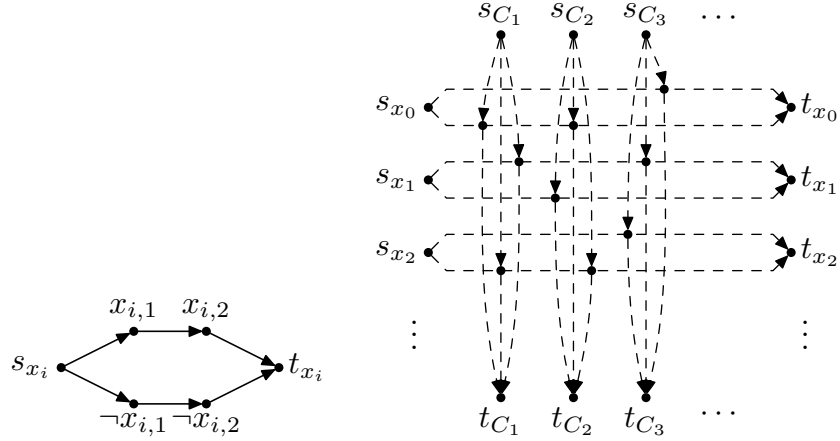


Figure 2: Left: gadget for variable  $x_i$ ; right: schematic of the construction.

constructed 2-SAT instance in polynomial time (Theorem 2.1). We omit the simple proof that the formula is satisfiable if and only if there is a solution to the  $k$ -PATH instance at hand.

On the other hand, given a digraph  $G$  with  $ddp(G) \leq 2$  and  $k$  pairs of nodes  $(s_1, t_1), \dots, (s_k, t_k)$ , we proceed as follows. By Corollary 3.12 d), every directed path in  $G$  has length at most 2. In the first step, for each pair  $(s_i, t_i)$  such that  $s_i = t_i$  or  $(s_i, t_i) \in E(G)$ , we simply remove both vertices  $s_i, t_i$  from the instance. Hence, for the second step of the algorithm, we may assume that every  $s_i-t_i$  path in  $G$ ,  $1 \leq i \leq k$ , is of length two, which means it is formed by a pair of arcs  $(s_i, x), (x, t_i) \in E(G)$ . We denote by  $X_i = \{x \in V(G) : (s_i, x), (x, t_i) \in E(G)\}$ , and define a bipartite graph  $B$  on the vertex set  $V(B) = V(G) \cup \{1, \dots, k\}$  by  $\{x, i\} \in E(B)$  if and only if  $x \in X_i$ . Then, clearly, the  $k$ -PATH instance has a solution if and only if  $B$  has a matching of size  $k$ , which can be decided in polynomial time.

b) Membership in NP is trivial. For the hardness part, we reduce from an NP-complete variant of the SAT problem described in Theorem 2.2. Our reduction is conceptually very similar to the one used already in [EIS76], but we pay specific attention to our new measures.

Let  $\varphi$  be a SAT-formula with  $m$  clauses  $C_1, \dots, C_m$  over  $n$  variables  $x_0, \dots, x_{n-1}$ , satisfying the conditions of Theorem 2.2. We create a digraph  $G_\varphi$  as follows. For every variable  $x_i$ , we add a new 6-vertex gadget as depicted in Figure 2 (left), and we call  $(s_{x_i}, x_{i,1}, x_{i,2}, t_{x_i})$  the *upper path* of  $x_i$  and  $(s_{x_i}, \neg x_{i,1}, \neg x_{i,2}, t_{x_i})$  the *lower path* of  $x_i$  in the gadget. For every clause  $C_j$ , we add two new vertices  $s_{C_j}$  and  $t_{C_j}$ , and connect them with appropriate variable-gadget vertices as follows.

For every literal  $\ell \in C_j$  such that this is the  $k$ -th occurrence of  $\ell$  within the formula ( $k \in \{1, 2\}$ ), we add the arcs  $(s_{C_j}, \ell_k)$  and  $(\ell_k, t_{C_j})$ . Note that the vertex, named here as  $\ell_k$ , is actually a vertex of the variable gadget correspond-

ing to  $\ell$ : if, for example,  $\ell = \neg x_5$  and  $\neg x_5$  has already occurred in some  $C_{j'}$  with  $j' < j$ , then  $k = 2$ ,  $\ell_2 = \neg x_{5,2}$  and so we add the edges  $(s_{C_j}, \neg x_{5,2})$  and  $(\neg x_{5,2}, t_{C_j})$ . See Figure 2 for a schematic overview. Briefly, choosing the upper path of  $x_i$  corresponds to assignment  $x_i = false$  since it leaves the nodes  $\neg x_{i,1}$  and  $\neg x_{i,2}$  available to satisfy their clauses, while the lower path corresponds to  $x_i = true$ .

It is easy to see that the resulting digraph is a DAG, the longest path has at most four vertices (which bounds the DAG-depth by Corollary 3.12), and between any two vertices there are at most three paths.

It remains to formally argue that  $\varphi$  is satisfiable if and only if  $G_\varphi$  is a “yes”-instance to the  $k$ -PATH problem with pairs  $(s_{x_i}, t_{x_i})$  for all  $0 \leq i < n$  and pairs  $(s_{C_j}, t_{C_j})$  for all  $1 \leq j \leq m$ . Consider a satisfying assignment for  $\varphi$ : We use the lower (upper) path for each pair  $(s_{x_i}, t_{x_i})$  if  $x_i = true$  (*false*), which leaves an available path  $(s_{C_j}, \ell_k, t_{C_j})$  for each clause  $C_j$  where  $\ell$  is the (one of) literal satisfying  $C_j$ . On the other hand, assume there is a solution to  $k$ -PATH on the constructed instance: There exist only two  $s_{x_i}$ - $t_{x_i}$  paths for each  $0 \leq i < n$ , and these determine the truth assignment for  $x_i$  as described above. Now, if a disjoint  $s_{C_j}$ - $t_{C_j}$  path in the solution uses a vertex  $\ell_k$  (where  $\ell \in C_j$ ), then our truth assignment is  $\ell = true$  and consequently each  $C_j$  is satisfied. ■

**Theorem 4.9.** *There is a fixed-parameter tractable algorithm solving the  $c$ -PATH problem (for fixed  $c$ ) on a digraph  $G$*

- a) *in time  $O(t^c \cdot |E(G)|)$  if  $G$  is of  $K$ -width at most  $t$ ;*
- b) *in time  $O((2c)^{ct^4} \cdot |E(G)|^2)$  if  $G$  is of DAG-depth at most  $t$ .*

*This holds also if  $t$  is unknown to the algorithm.*

**Proof:** Notice that we can, without loss of generality of the  $c$ -PATH problem, assume that  $G$  has no parallel arcs (while 2-cycles are permitted).

a) We, for each  $i = 1, \dots, c$ , apply Proposition 3.15 to list all  $\leq t$  distinct directed paths from  $s_i$  to  $t_i$ . Then, using brute force over all  $t^c$  possibilities, we check whether there is a selection of pairwise disjoint ones.

b) This algorithm uses part (a) and recursive calls. By Corollary 3.12, the longest directed path in  $G$  has length  $\ell < 2^t$  (which is the only extra information we use about  $G$ ). We are actually going to solve a more general *constrained*  $c$ -PATH problem: given  $G$ , the pairs  $(s_i, t_i)$  and sets  $E_i \subseteq E(G)$ ,  $i = 1, \dots, c$ , the task is to find a collection of  $c$  pairwise disjoint directed  $s_i$ - $t_i$  paths  $Q_i$  in  $G$  such that  $E(Q_i) \subseteq E_i \subseteq E(G)$ . Initially,  $E_1 = \dots = E_c = E(G)$ .

Let  $\mathcal{P}_i$  be the collection of all  $s_i$ - $t_i$  paths with arcs from  $E_i$ . If  $|\mathcal{P}_i| \leq (c\ell)^{\ell^2}$  for all  $i = 1, \dots, c$ , then we may actually use (a) to solve the problem in time  $O((c\ell)^{c\ell^2} \cdot |E(G)|)$ . Otherwise,  $|\mathcal{P}_i| > (c\ell)^{\ell^2}$  for some  $i$ , and we may apply the following for  $u = s_i$ ,  $v = t_i$ , and  $m = c\ell$ ,  $k = \ell$ :

**Claim 1.** *Let  $H$  be a simple digraph, and  $u, v$  two vertices of  $H$  such that the longest path starting in  $u$  has length  $k + 1$  and there exist  $1 + (m - 1)^{k^2}$  distinct*

directed  $u$ - $v$  paths in  $H$ . Then there exist vertices  $u', v'$  in  $H$  such that there are  $m$  pairwise internally disjoint  $u'$ - $v'$  paths.

To prove the claim, we may assume that every arc of  $H$  is on some  $u$ - $v$  path. By the pigeon-hole principle, there exists a vertex  $u'$  in  $H$  having outdegree  $\geq 1 + (m-1)^k$ , and this  $u'$  is not an in-neighbour of  $v$  (otherwise, we would have only  $\leq ((m-1)^k)^k$  distinct  $u$ - $v$  paths). Let  $u'_i, i = 1, \dots, p \geq 1 + (m-1)^k$  be the out-neighbours of  $u'$  in  $H$ , and let  $H'$  be an inclusion-wise minimal subgraph of  $H$  such that  $H'$  contains a  $u'$ - $v$  path passing through  $u'_i$  for all  $i = 1, \dots, p$ . By a symmetric application of the pigeon-hole principle, there exists a vertex  $v'$  having indegree  $\geq m$  in  $H'$ . Let  $v'_j, j = 1, \dots, q \geq m$  be the in-neighbours of  $v'$  in  $H'$ . By minimality of  $H'$  there exists a  $u'$ - $v'$  path  $R_j$  through each  $v'_j$ ; we claim that these paths are pairwise internally disjoint. If, say,  $X = (V(R_1) \cap V(R_2)) \setminus \{u', v'\} \neq \emptyset$ , we could choose  $x \in X$  such that the arcs  $f_1 \in E(R_1), f_2 \in E(R_2)$  leaving  $x$  are distinct, but then  $H' - f_2$  would witness that  $H'$  was not minimal.

In other words, there exist vertices  $s', t'$  in  $G$  such that  $c\ell$  suitable fragments of paths from  $\mathcal{P}_i$  form pairwise internally disjoint  $s'$ - $t'$  paths  $R_1, \dots, R_{c\ell}$ . These paths can be found in time  $O((c\ell)^{\ell^2} \cdot |E(G)|)$  using an approach similar to Proposition 3.15 a). Now, we make a new arc set  $E'_i$  from  $E_i$  by removing all arcs of  $R_1 \cup \dots \cup R_{c\ell}$ , and adding a new arc  $f' = (s', t')$ . We call the same algorithm recursively with the constraints  $E_1, \dots, E'_i, \dots, E_c$ .

This algorithm clearly stops after  $O(c|E(G)|)$  recursive calls since each call decreases  $|E_1| + \dots + |E_c|$ . Hence the overall run-time is  $O((c\ell)^{c\ell^2} \cdot |E(G)|^2)$ . It remains to prove that there is a solution with constraints to  $E_1, \dots, E_i, \dots, E_c$  if and only if there is a solution to  $E_1, \dots, E'_i, \dots, E_c$ . The “only if” direction is trivial since we can simply use the arc  $f' = (s', t')$  when needed.

In the “if” direction, when  $f'$  is not used in the path, we are done. If  $f'$  is used in the  $s_i$ - $t_i$  path  $Q'_i$ , then we notice the following: by the pigeon-hole principle, some of the paths  $R_j \in \{R_1, \dots, R_{c\ell}\}$  must be disjoint from all other  $c-1$  paths of  $\leq \ell$  vertices in the solution, and hence we can use the path  $(Q_i - f') \cup R_j$  with all arcs in  $E_i$  instead. ■

#### 4.4. Max-/Min-Leaf Outbranching (MAXLOB, MINLOB)

An *outbranching* in a digraph  $G$  is a spanning rooted tree  $T$  of  $G$  such that all the arcs of  $T$  are oriented away from its root. An outbranching need not always exist, but it is easy to check for its existence using the depth-first search from each vertex of  $G$ .

The *Max-leaf (Min-leaf) outbranching* problem is to find an outbranching  $T$  in a digraph  $G$  such that the number of leaves of  $T$  is maximized (minimized, respectively). We denote the problems by MAXLOB and MINLOB, respectively. We also consider the  $c$ -MINLOB problem asking for an existence of an outbranching with at most  $c$  leaves, where  $c$  is a fixed problem constant.

Note that the two similarly defined problems may have very different algorithmic behaviour. Already the problem 1-MINLOB (equivalent to a Hamiltonian path) is NP-complete. On the other hand, analogous “1-MAXLOB”

problem is trivial (a nonempty outbranching always has a leaf), and the problem whether a given digraph  $G$  has an outbranching with at least  $k$  leaves (“ $k$ -MAXLOB”) is FPT when parameterized by  $k$  [KLR08]. For more results on the MAXLOB problem we therefore refer to Section 4.5 where a related dominating set problem is studied, and here we stay with MINLOB variants.

**Theorem 4.10.** *The MINLOB problem;*

- a) *can be solved in polynomial time on DAGs (Gutin, Razgon, and Kim [GRK09]), but is NP-hard on digraphs of directed path-width 1, cycle rank 1, and DAG-width 2 (Dankelmann, Gutin, and Kim [DGK09]<sup>1</sup>).*
- b) *has an XP algorithm with respect to clique-width ([GHO11]).*

*The  $c$ -MINLOB problem has an XP algorithm with respect to DAG-width ([DGK09]) and clique-width ([GHO13]).*

With the new width parameters we get a positive and a negative result.

**Proposition 4.11.** *There is a fixed-parameter tractable algorithm solving the  $c$ -MINLOB problem (for any fixed  $c$ ) on a digraph  $G$*

- a) *in time  $2^{O(c^2 2^{2k})}$  if  $G$  is of DAG-depth at most  $k$  ( $k$  must be known to the algorithm);*
- b) *in time  $O(|V(G)|^{c+3} \cdot ct^c)$  if  $G$  is of  $K$ -width  $t$  ( $t$  not necessarily known).*

**Proof:** a) Let  $G$  be of DAG-depth  $k$ . By Corollary 3.12, the longest path length in  $G$  is at most  $2^k - 2$ , and so an outbranching with  $c$  leaves could cover at most  $1 + (2^k - 2)c < 2^k c$  vertices of  $G$ . If  $|V(G)| \geq 2^k c$ , then we immediately answer No. Otherwise, we enumerate by brute force all possible arc subsets and check whether one of them is an out-branching with at most  $c$  leaves.

b) We exhaustively loop through all  $O(|V(G)|^{c+1})$  selections of the root and the  $c$  leaves in  $G$ . Using Proposition 3.15 we enumerate all the  $\leq ct$  paths from the chosen root to each leaf in time  $O(ct \cdot |V(G)|^2)$ , and then we simply check whether some of the  $t^c$  possible selections of  $c$  of these paths covers the whole graph  $G$ . ■

**Theorem 4.12.** *The MINLOB problem is NP-hard on digraphs*

- a) *that are of DAG-depth 5 and  $K$ -width 6;*
- b) *having a directed feedback vertex set of size one.*

---

<sup>1</sup>Note that [DGK09] use a different definition of DAG-width with the value by one lower than ours.

Especially part b) is noteworthy in comparison to the positive results in Theorem 4.10.

**Proof:** a) We again use the variant of SAT described in Theorem 2.2: Let  $\varphi$  be a SAT-formula with  $m$  clauses  $C_1, \dots, C_m$  over  $n$  variables  $x_1, \dots, x_n$ , satisfying the conditions of Theorem 2.2.

We construct a digraph  $G_\varphi$  from a single source vertex  $s$ , a disjoint copy  $H_i$  of the gadget from Fig. 3 for each variable  $x_i$ ,  $i = 1, \dots, n$ , and sink vertices  $c'_j$  for each clause  $C_j$ ,  $j = 1, \dots, m$ . The gadget  $H_i$  consists of five vertices  $x'_i, x''_i, h_i, \neg x'_i, \neg x''_i$ , out of which  $x'_i, \neg x'_i$  have arcs from  $s$ . For each clause  $C_j$  containing the positive literal  $x_i$ , there is an arc from either  $x'_i$  or  $x''_i$  (depending on whether this is its first or second occurrence within  $\varphi$ ) to  $c'_j$ . Likewise for  $C_{j'}$  containing negative  $\neg x_i$ , there is an arc from either  $\neg x'_i$  or  $\neg x''_i$  to  $c'_{j'}$ .

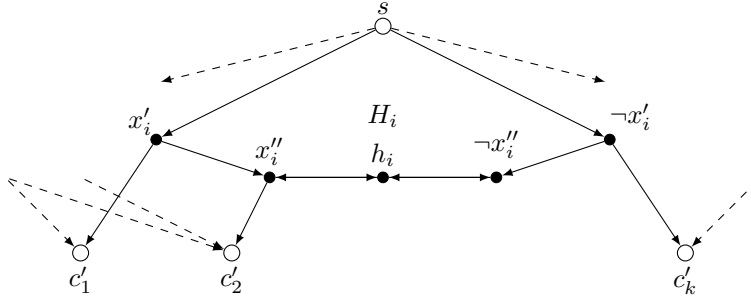


Figure 3: The MINLOB reduction gadget for a variable  $x_i$  (in the black vertices); with schematic connections from the common source  $s$  and to gadget sinks  $c'_j$ .

The claim is that  $\varphi$  is satisfiable iff  $G_\varphi$  has a min-leaf outbranching with exactly  $2n$  leaves where  $n$  is the number of variables in  $\varphi$ . First assume that  $\varphi$  is satisfiable. We cover the vertices of the gadget  $H_i$  with two directed paths  $(s, x'_i)$  and  $(s, \neg x'_i, \neg x''_i, h_i, x''_i)$  if  $x_i$  is true in the satisfying assignment of  $\varphi$ , or with  $(s, \neg x'_i)$  and  $(s, x'_i, x''_i, h_i, \neg x''_i)$  otherwise. This sums up to  $2n$  paths starting in  $s$ . For each  $j = 1, \dots, m$ , we choose a true literal  $\ell \in C_j$ , and prolong the path ending so far in  $\ell'$  or  $\ell''$  to cover also the sink  $c'_j$ .

Conversely, assume that  $G_\varphi$  has a min-leaf outbranching  $T$  with exactly  $2n$  leaves. Since  $s$  is a unique source in the graph,  $s$  must be the root. Then the  $2n$  out-neighbours of  $s$  form an independent set reachable only from  $s$ , and so each one must be covered by a distinct path in  $T$ . Hence  $T$  consists of exactly  $2n$  paths  $P_i, P'_i$  starting in  $s$  and pairwise disjoint otherwise, such that  $P_i \cup P'_i$  covers  $V(H_i)$  for  $i = 1, \dots, n$ . There are only two symmetrical possibilities for  $P_i \cap V(H_i)$  and  $P'_i \cap V(H_i)$ , as described in the previous paragraph, since one of the paths must pass through  $h_i$ . We assign  $x_i = \text{true}$  if both  $h_i$  and  $\neg x'_i$  belong to the same path among  $P_i, P'_i$ , and  $x_i = \text{false}$  otherwise. Now, for each  $j = 1, \dots, m$ , the sink vertex  $c'_j$  must end some of the paths  $P_i$  or  $P'_i$  (since  $c'_j$  also belongs to  $T$ ), and then the corresponding literal  $x_i$  or  $\neg x_i$  in  $C_j$  has been evaluated true in our assignment.

It remains to verify that  $G_\varphi$  has DAG-depth 5 and K-width 6. Regarding the

former, we show a lift-free search strategy for 5 cops (Theorem 3.11, which gives a stronger conclusion than considering a longest directed path on 6 vertices) in  $G_\varphi$ : the first cop landing on  $s$  and the second one on respective  $h_i$ , leave the robber isolated at  $x'_i, x''_i$  and the adjacent (at most two) sink vertices. Then the remaining three cops catch the robber. It is similarly straightforward to verify that no two vertices of  $G_\varphi$  are connected by more than six distinct paths (an equality occurring between  $s$  and a sink vertex).

b) We reduce from the same problem as in a), using similar ideas but different gadgets for variables of  $\varphi$  – see Fig. 4 where the variable gadgets form a “chain of hexagons”. Let the copies of the variable gadget in  $G_\varphi$  be again denoted by  $H_1, H_2, \dots, H_n$  where  $H_i$  and  $H_{i+1}$  share a vertex.

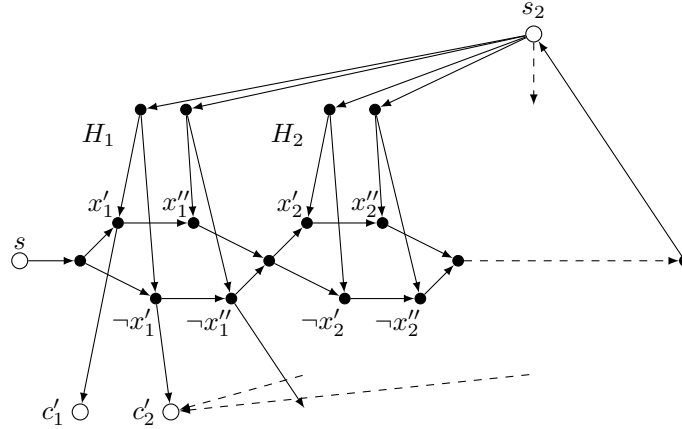


Figure 4: Alternative MINLOB reduction gadgets for variables  $x_1, x_2, \dots$  (black vertices).

Assuming a given satisfying assignment for  $\varphi$ , we again easily produce an outbranching in  $G_\varphi$  with  $2n$  leaves: start with a single directed path  $Q$  from  $s$  to  $s_2$  (taking the “upper” section of each  $H_i$  in Fig. 4 if  $x_i$  is false, and the “lower” section otherwise), then branch twice to each variable gadget, and finally use an arbitrary true literal of each clause to cover the sink vertices  $c'_1, c'_2, \dots$ .

On the other hand, assume  $T$  is an outbranching in  $G_\varphi$  with  $2n$  leaves. Then  $T$  must contain a directed  $s$ – $s_2$  path  $Q$ . Since the  $2n$  out-neighbours of  $s$  are reachable from  $s$  only through  $s_2$ ,  $T$  has to branch to (at least, but also at most)  $2n$  paths starting from  $s_2$  which we denote by  $P_i, P'_i$ ,  $i = 1, \dots, n$ , according to the gadgets  $H_i$  they meet. Since  $T$  has  $2n$  leaves, it must be that  $T = Q \cup \bigcup_i (P_i \cup P'_i)$ . We assign  $x_i = \text{true}$  if  $Q$  meets  $H_i$  in  $\neg x'_i, \neg x''_i$ , and  $x_i = \text{false}$  otherwise. Now, again, for each  $j = 1, \dots, m$ , the sink vertex  $c'_j$  must end some of the paths  $P_i$  or  $P'_i$  in  $T$ , and then the corresponding literal  $x_i$  or  $\neg x_i$  in  $C_j$  has been evaluated true in our assignment.

To conclude the proof, note that  $G_\varphi - s_2$  is a DAG. ■

4.5. *Directed Dominating Set (DiDS) and Directed Steiner Tree (DiST)*

The *Directed Dominating Set* problem (DiDS) asks for a minimum cardinality vertex set  $X$  in a digraph  $G$  such that every vertex of  $G$  not in  $X$  is an out-neighbour of some vertex of  $X$  (in other words,  $X$  dominates all vertices in  $G$ ). The (*unit-cost*) *Directed Steiner Tree* problem (DiST) [HRW92], given a digraph  $G$  and  $T \subseteq V(G)$ ,  $r \in V(G)$ , asks for a minimum size tree in  $G$  spanning  $\{r\} \cup T$  with all arcs oriented away from  $r$ . While it is folklore that both of these problems are NP-hard in general, we show with a simple reduction that the same holds even on very restricted graph classes (and applies to the aforementioned MAXLOB, too).

**Theorem 4.13.** *The DiDS, DiST, and MAXLOB problems are NP-complete on DAGs of K-width 2 and DAG-depth 3.*

**Proof:** We use a reduction from the classical *vertex cover* (VC) problem to show hardness. Let a graph  $G = (V, E)$  and  $k \in \mathbb{N}$  be an input instance for VC. Without loss of generality we can assume that  $|V| \geq k + 2$ .

We construct  $G' = (V', E')$  as follows. Let  $V' = V \cup E \cup \{v_0\}$  and

$$E' = \{(v_0, v) : v \in V\} \cup \{(v, e) \in V \times E : v \in e\}.$$

We show that  $G = (V, E)$  has a vertex cover of size  $k$  if and only if  $G' = (V', E')$  has a directed dominating set of size  $k + 1$ : Assume that there is some vertex cover  $C \subseteq V$  of size  $k$  in  $G$ . Then  $\{v_0\} \cup C$  is a directed dominating set in  $G'$ , because  $v_0$  dominates itself as well as all  $v \in V$ , and since each  $e \in E$  is incident to some  $v \in C$ ,  $C$  dominates  $E$  in  $G'$ . Now let  $D$  be a directed dominating set in  $G'$  of size  $k + 1$ . It is  $v_0 \in D$  since otherwise some node in  $V$ ,  $|V| \geq k + 2$ , would not be dominated. Moreover, we can assume that  $D \cap E = \emptyset$ , because each  $e \in E$  can only dominate itself in  $G'$ . It is thus always safe to pick a predecessor of  $e$  instead. But then, each  $e \in E$  is dominated by some  $v \in D \cap V$ , and thus  $D \cap V$  is a vertex cover in  $G$ .

Finally,  $G'$  is a DAG with K-width two, since there are only two paths from  $v_0$  to each  $e \in E$ , only one path from  $v_0$  to each  $v \in V$  and only one path from each  $v \in V$  to each  $e \in E$ . Likewise, the DAG-depth of  $G'$  is at most three by Corollary 3.12.

The same construction can also be used to prove hardness for the other two problems. We claim that our  $G'$  has a directed dominating set  $D$  of size  $k + 1$  if and only if  $G'$  has a Steiner tree  $S$  of size  $k + 1 + |E|$  with the root  $r = v_0$  and the terminal set  $T = E$ . In the “if” direction,  $v_0 \in V(S)$  dominates all vertices in  $V$  and the  $k$  vertices of  $V(S) \cap V$  dominate whole  $E$  by the definition of  $S$ . Conversely,  $v_0$  must belong to any directed dominating set  $D$  of size  $k + 1$ , and so by adding  $\leq |E|$  arcs with heads in  $E$  into  $G'[D]$  one gets a Steiner tree from  $v_0$  to  $T = E$ .

Moreover, we show that  $G'$  has a directed dominating set  $D$  of size  $k + 1$  if and only if  $G'$  has an outbranching with  $\geq |V| + |E| - k$  leaves: Let  $D$  be such a dominating set. As argued above,  $v_0 \in D$  and it can be assumed  $D \cap E = \emptyset$ .

Hence,  $D$  induces a start rooted at  $v_0$  and all vertices of  $V(G') \setminus D$  thus can be leaves of an outbranching from  $v_0$ . Conversely, assume an outbranching  $U \subseteq G'$  with  $\ell \geq |V| + |E| - k$  leaves. Then all  $|V(G')| - \ell \leq k + 1$  non-leaf vertices of  $U$  form a directed dominating set by the definition. ■

**Proposition 4.14.** *The DiDS, DiST, and MAXLOB problems can be formulated as LinEMSO<sub>1</sub> optimization problems, and hence DiDS, DiST, and MAXLOB are fixed-parameter tractable with respect to clique-width.*

**Proof:** Let  $G$  be a digraph. From Example 4.1 we have a formula

$$\delta(X) \equiv \forall z(z \in X \vee \exists x \in X. \text{arc}(x, z))$$

recognizing a dominating set  $X$  in  $G$ , and hence an LinEMSO<sub>1</sub> optimization problem

$$\min \{ |X| : X \subseteq V(G) \text{ and } G \models \delta(X) \}.$$

Then DiDS is in FPT for the parameter clique-width by Theorem 4.2.

Second, let  $T \subseteq V(G)$  and  $r \in V(G) \setminus T$  in a DiST instance. We reformulate the problem as to minimize the cardinality of  $X \subseteq V(G)$  such that  $X$  induces in  $G$  directed paths from  $r$  to all the vertices of  $T$ . We can thus write (with constants  $r$  and  $T$ )

$$\begin{aligned} \sigma(r, T, X) \equiv & r \in X \wedge T \subseteq X \wedge \\ & \forall t \in T \forall Z \subseteq X ((r \in Z \wedge t \notin Z) \rightarrow \exists x \in Z, y \in X \setminus Z. \text{arc}(x, y)) \end{aligned}$$

and then

$$\min \{ |X| : X \subseteq V(G) \text{ and } G \models \sigma(r, T, X) \}.$$

We finish by Theorem 4.2.

Third, we reformulate the MAXLOB problem as to minimize the set  $Y$  of non-leaf vertices in an outbranching of  $G$ . Recycling the MAXLOB-related argument in the proof of Theorem 4.13, we obtain a LinEMSO<sub>1</sub> formulation

$$\min \{ |Y| : Y \subseteq V(G) \text{ and } G \models \delta(Y) \wedge \exists r \in Y. \sigma(r, Y, Y) \}.$$

We again finish by Theorem 4.2. ■

#### 4.6. Maximum cardinality Directed Cut (MAXDicut)

The *Maximum cardinality directed cut* (MAXDicut) problem is defined as follows: given a digraph  $G$ , partition the vertex set  $V(G)$  into  $V_0$  and  $V_1$  such that the cardinality of  $\{(u, v) \in E(G) : u \in V_0, v \in V_1\}$  is maximized.

**Theorem 4.15.** *The MAXDicut problem*

- a) *is NP-hard on DAGs (Lampis, Kaouri, and Mitsou [LKM11]);*
- b) *has an XP algorithm ([GHO13]) but is W[1]-hard when parameterized by clique-width (Fomin, Golovach, Lokshantov, and Saurabh [FGLS10a]).*



A closer, yet nontrivial, look at the reduction in [LKM11] reveals that the resulting graph also has bounded DAG-depth and K-width:

**Theorem 4.16 (implicit in [LKM11]).** *The MAXDicut problem is NP-hard on DAGs of K-width at most 4608 and DAG-depth at most 5.*

**Proof:** To show that the [LKM11, Theorem 3] reduction from so-called not-all-equal 3-SAT has the required additional properties, we argue as follows.

Let a formula  $\varphi$  be an input CNF instance with variables  $x_1, \dots, x_n$ . [LKM11] argue (and use) that the following assumption can be made about  $\varphi$ , without loss of generality: each clause of  $\varphi$  has exactly 3 literals, and each variable occurs as positive and negative literal the same number of times. Let  $occ(x)$  stand for the total number of occurrences of a variable  $x$  in  $\varphi$ . We claim that, in addition to the aforementioned assumptions, one can assume  $occ(x) \leq 4$  for each variable  $x$  in  $\varphi$ : If, say,  $occ(x) = 2k > 4$ , then we add  $2k + 2$  new variables  $y_0, \dots, y_k, y'_0, \dots, y'_k$  and the new clauses  $(y_1 \vee \neg y_2 \vee \neg y_2) \wedge \dots \wedge (y_k \vee \neg y_0 \vee \neg y_0) \wedge (y_0 \vee y_0 \vee \neg y'_1) \wedge (y'_1 \vee y'_1 \vee \neg y_2) \wedge \dots \wedge (y'_k \vee y'_k \vee \neg y'_0) \wedge (y'_0 \vee \neg y_1 \vee \neg y_1)$ , and replace the  $k$  positive literals of  $x$  with  $y_1, \dots, y_k$  and the  $k$  negative ones with  $\neg y'_1, \dots, \neg y'_k$ . After finishing this construction for each violating  $x$  we arrive at a formula satisfying all the assumption at once and equivalent to the former one in not-all-equal 3-SAT.

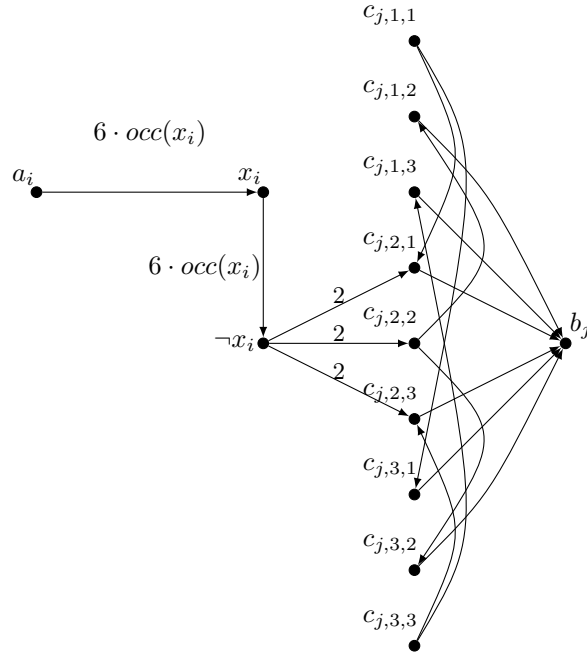


Figure 5: Weighted MaxDiCut reduction for Theorem 4.16; a clause- $C_j$  gadget with the literal ‘ $\neg x_i$ ’ occurring at the second position in  $C_j$ . Arc weights are 1 except the denoted arcs incident with  $x_i, \neg x_i$ .

It remains to analyze the graph  $G_\varphi$  that [LKM11] construct for such  $\varphi$  in their reduction. We give a detail of the weighted construction [LKM11] featuring one clause and one literal gadgets in Figure 5, which is enough for our purpose. To obtain an unweighted graph  $G_\varphi$  the construction from [LKM11, Theorem 4] is subsequently used. This construction first replaces each arc of weight  $k$  by  $k$  parallel arcs, and then replaces each parallel arc by a directed path of length 3 (two new vertices are added for each such arc).

$G_\varphi$  is of DAG-depth 5 by Theorem 3.11 and the following search strategy: At any game position the (visible) robber can reach only one of the sets  $\{a_i, x_i, \neg x_i\}$ ,  $i = 1, \dots, n$ , and so the cop player lands on the respective vertices  $x_i$  and  $\neg x_i$  in his first two moves. If the robber does not escape through  $x_i$  or  $\neg x_i$ , then he is easily caught in subsequent three moves. Otherwise, the third cop lands on respective  $c_{j,k,k'}$  (through which the robber escaped), and at most two more moves are enough for him to win as can be seen from Figure 5. To compute K-width we notice that the highest number of paths occurs between some  $a_i$  and  $b_j$  and can be at most  $6 \text{occ}(x_i) \cdot 6 \text{occ}(x_i) \cdot 2(1+2+1) \leq (6 \cdot 4)^2 \cdot 2 \cdot 4 = 4608$ . Hence the reduction of [LKM11] proves hardness even in the case of DAG-depth  $\leq 5$  and K-width  $\leq 4608$ . ■

#### 4.7. Oriented Colouring ( $c$ -OCN)

A possible directed generalization of the ordinary graph colouring problem can be obtained as follows. A *graph homomorphism* is a mapping  $V(G) \rightarrow V(H)$  that maps edges (arcs) of  $G$  to edges (arcs) of  $H$ . The chromatic number  $\chi(G)$  of a graph  $G$  then equals the minimum  $c$  such that  $G$  has a homomorphism into the (loopless) complete graph  $K_c$ . For digraphs  $G$ , then, the *Oriented Chromatic Number* (OCN)  $\chi_o(G)$  equals the minimum integer  $c$  such that  $G$  has a homomorphism into some(!) orientation of  $K_c$ , where *orientation* of an undirected graph  $H$  is a digraph having exactly one of the arcs  $(u, v)$  or  $(v, u)$  for every edge  $uv$  of  $H$ .

In other words,  $\chi_o(G)$  equals the minimum  $c$  such that the vertex set of  $G$  can be partitioned into  $c$  independent (arc-free) sets such that, between each pair of the sets, all arcs have the same direction. For instance,  $\chi_o(\vec{C}_5) = 5$ . We denote by  $c$ -OCN the problem to test whether  $\chi_o(G) \leq c$  where  $c$  is a problem constant.

**Theorem 4.17.** *a) There is a polynomial algorithm for 3-OCN on all digraphs (Klostermeyer and MacGillivray [KM04]).*

*b) The problem 4-OCN is NP-complete on DAGs (Culus and Demange [CD06]).*

A simpler and more powerful reduction than [CD06] strengthens the hardness conclusion as follows:

**Theorem 4.18.** *The problem 4-OCN is NP-complete on*

*a) DAGs of K-width 3 and DAG-depth 5;*

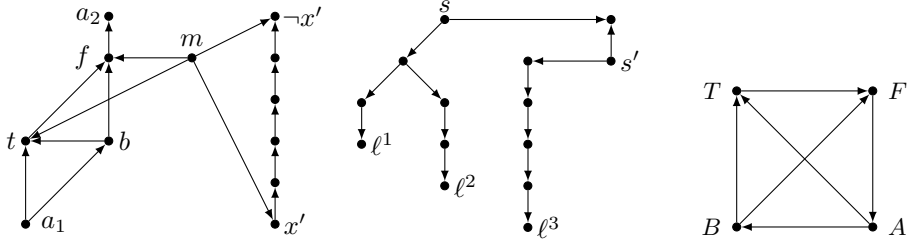


Figure 6: Gadgets  $L$  (left),  $S$  (middle), and  $K$  (right), for the proof of Theorem 4.18.

b) digraphs of DAG-width 2,  $K$ -width 1 and DAG-depth 3.

**Proof:** a) Let  $L, S$  and  $K$  be the three gadgets from Figure 6. We claim the following properties:

- i. There is a homomorphism of  $L$  to  $K$  such that the pair  $(x, \neg x)$  is mapped to  $(T, F)$ , and another one that maps  $(x, \neg x)$  to  $(F, T)$  (Figure 7).
- ii. For each triple  $\tau \in \{T, F\}^3$  except  $\tau = (F, F, F)$ , there is a homomorphism of  $S$  to  $K$  such that the triple  $(\ell^1, \ell^2, \ell^3)$  is mapped to  $\tau$  (Figure 8).

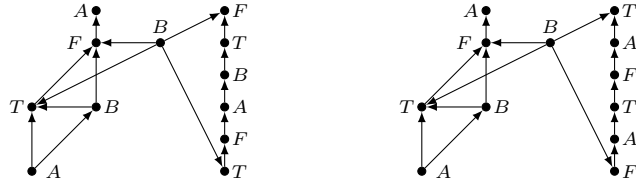


Figure 7: Two interesting homomorphisms of  $L$  to  $K$  (cf. Figure 6)—the target vertices are written in the picture.

$(l_1, l_2, l_3) \rightarrow$ :	$s-l_1$	$s-l_2$	$s-s'-l_3$
$(T, T, T)$	BFAT	BFABT	BFBTFABT
$(T, T, F)$	BFAT	BFABT	BFBTFABF
$(T, F, T)$	BFAT	BFABF	BFBTFABT
$(T, F, F)$	BFAT	BFABF	BFBTFABF
$(F, T, T)$	ABTF	ABFAT	ABABFABT
$(F, T, F)$	ABTF	ABFAT	ABABFABF
$(F, F, T)$	FABF	FABTF	FAFABFAT

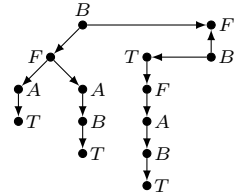


Figure 8: Seven interesting homomorphisms of the gadget  $S$  to  $K$ , where the table shows the images of each of the three (undirected) paths  $s-l^1, s-l^2, s-l^3$  of  $S$ . The picture to the right features the mapping given in the first row.

Our reduction works as follows: Given a CNF formula  $\varphi$  satisfying the conditions of Theorem 2.2, for every variable  $x_i$  we construct a copy  $L_i$  of the

gadget  $L$  from Figure 6, and note its vertices  $x'_i, \neg x'_i$  representing the literals  $x_i, \neg x_i$ . For every clause  $C_j$  we construct a copy  $S_j$  of the gadget  $S$ , where its vertices  $\ell_j^1, \ell_j^2$  and  $\ell_j^3$  are identified with the vertices we have created for the literals appearing in  $C_j$ . Then (i),(ii) guarantee that the resulting digraph  $G_\varphi$  has a homomorphism to  $K$  (an orientation of  $K_4$ ) if  $\varphi$  is satisfiable.

On the other hand, assume we are given a homomorphism of  $G_\varphi$  to some orientation  $K'$  of  $K_4$ . The vertices  $a_1, t, b, f$  of the  $L$  gadget (each copy of it, as follows from the coming argument) need to be pairwise distinct vertices of  $K'$ , and without loss of generality these colours are named  $A, T, B, F$ , respectively. This determines the orientation of all edges of  $K'$  except  $AF$ , which then must be  $(F, A)$  since otherwise there would be no room to map the arc  $(f, a_2)$  of  $L$ . Consequently,  $K' \simeq K$  and each  $L_i$  is mapped to  $K'$  in the same way.

Notice that  $a_2$  and  $m$  must then be mapped to  $A$  and  $B$ , respectively, and  $x', \neg x'$  into  $\{T, F\}$ . However, the directed  $x' - \neg x'$  path of length 5 in  $L$  admits neither a homomorphism with  $x', \neg x' \rightarrow T$  nor with  $x', \neg x' \rightarrow F$ , as can be easily checked from the picture. The vertex pair  $(x'_i, \neg x'_i)$  in each  $L_i$  thus has to be mapped to either  $(T, F)$  or  $(F, T)$ , which we naturally use to assign the logical value to  $x_i$ . Finally, one can exhaustively verify that there exists no homomorphism of  $S$  to  $K'$  that maps  $(\ell^1, \ell^2, \ell^3)$  to  $(F, F, F)$ , and hence  $\varphi$  is satisfiable by our assignment.

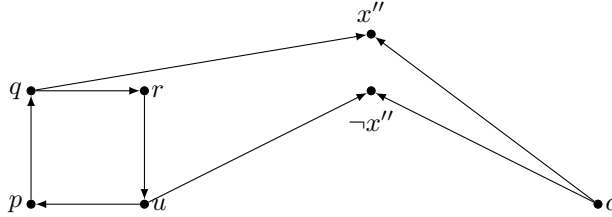


Figure 9: Gadget  $L'$

b) We proceed in the same manner as in a), but instead of the gadget  $L$  we utilize  $L'$  from Figure 9. For that we again argue that any homomorphism of  $L'$  to some orientation  $K'$  of  $K_4$  forces the images of  $p, q, r, u$  to be distinct and so forming a directed 4-cycle in  $K'$ . Already this fact certifies  $K' \simeq K$ , and hence the images of  $x''_i, \neg x''_i$  are distinct. Then  $c \rightarrow \{A, B\}$  and finally  $(x''_i, \neg x''_i) \rightarrow (T, F)$  or  $(x''_i, \neg x''_i) \rightarrow (F, T)$ . The rest follows as in a). ■

**Proposition 4.19.** *For every integer  $c$ , the problem  $c$ -OCN is fixed-parameter tractable with respect to clique-width.*

**Proof:** We write an MSO<sub>1</sub> formula

$$\exists X_1, \dots, X_c \left[ \bigwedge_{i=1, \dots, c} \forall x, y \in X_i (\neg \text{arc}(x, y)) \right. \\ \left. \wedge \bigwedge_{i, j=1, \dots, c} \forall x, y \in X_i, z, t \in X_j (\text{arc}(x, z) \rightarrow \neg \text{arc}(t, y)) \right]$$

which deals with the sets  $X_i$  of vertices of  $G$  that are mapped to the vertex  $i$  of (an orientation of)  $K_c$ . Hence the result follows from Theorem 4.2. ■

#### 4.8. Directed Feedback Vertex Set (DFVS) and Kernel (KERNEL)

The *directed feedback vertex set* (DFVS) optimization problem (mentioned already as a width parameter  $\text{dfn}(G)$  in Definition 3.2) is to find the minimum cardinality of a set  $S$  of vertices of a digraph  $G$  whose removal leaves  $G \setminus S$  acyclic. A *kernel* of a digraph  $G$  is defined as an independent set  $R \subseteq V(G)$  such that for every  $x \in V(G) \setminus R$  there is an arc from  $x$  into  $R$ . Notice that a kernel may not always exist, and the KERNEL decision problem is to find out whether an input digraph  $G$  has a kernel.

These two are again NP-hard problems, but they become trivial on DAGs. Yet no complexity improvement seems possible on them when using width parameters other than clique-width:

**Theorem 4.20.** *Let  $G$  be a digraph.*

- a) *The KERNEL problem is NP-complete even if  $G$  has DAG-width and K-width 2, cycle rank also 2, and DAG-depth 4 (van Leeuwen [vL76]).*
- b) *The DFVS problem is NP-hard even if  $G$  has DAG-width 4 and cycle rank 4 (Kreutzer and Ordyniak [KO11]).*
- c) *Computing  $\text{dfn}(G)$  is in FPT with the parameter  $\text{dfn}(G)$  (Chen, Liu, Lu, O’Sullivan, and Razgon [CLL<sup>+</sup>08]).*
- d) *The KERNEL and DFVS problems can be formulated in the LinEMSO<sub>1</sub> framework, and hence KERNEL and DFVS are in FPT when parameterized by clique-width (Example 4.1).*

Note that the reduction in claim b) produces digraphs of arbitrarily high DAG-depth and K-width, and so the parameterized complexity of DFVS parameterized by DAG-depth and K-width remains open.

**Theorem 4.21.** *If a digraph  $G$  is given with a directed feedback vertex set of size  $k$ , then the KERNEL problem can be solved in FPT time  $O(2^k \cdot (|V(G)| + |E(G)|))$ .*

**Proof:** Recall a simple folklore fact; if  $H$  is a DAG, then  $H$  has a unique kernel  $Z$  which can be easily computed as follows

- add to  $Z$  all the sink vertices (with no out-neighbour) of  $H$ ,
- denote by  $N = N^-[Z]$  the set of all in-neighbours of  $Z$  in  $H$ , and continue in the first step with the subdigraph  $H - (Z \cup N)$ .

We now consider an arbitrary digraph  $G$  with a feedback set  $S \subseteq V(G)$  of size  $k$ , and a (yet unknown) kernel  $Z \subseteq V(G)$ . If we knew the intersection  $Z \cap S$ , then we would uniquely determine the remainder  $Z \setminus S$  by calling the previous procedure on the DAG  $G - (S \cup N^-[Z \cap S])$ . This leads to a simple FPT algorithm:

- Enumerate all independent subsets  $Z_0 \subseteq S$  in  $O(2^k)$  rounds.
- For each  $Z_0$  from (i.), compute the unique kernel  $Z_1$  of  $G - (S \cup N^-[Z_0])$  in  $O(|V(G)| + |E(G)|)$  time.
- Check whether  $Z_0 \cup Z_1$  is a kernel of whole  $G$ .

If  $G$  has a kernel  $Z$  then the iteration of  $Z_0 = Z \cap S$  succeeds, thanks to uniqueness of the kernel of  $G - (S \cup N^-[Z_0])$ , while if  $G$  has no kernel then all the rounds clearly fail. ■

#### 4.9. Parity Games (PARITY)

*Parity games* play an important role in the field of model-checking and formal verification. The problem of solving, i.e., determining the winning player for parity games is equivalent to model checking modal  $\mu$ -calculus, an important modal logic subsuming many other logics. However, the exact complexity of solving parity games is a long-standing open problem. It is known to be in  $\text{NP} \cap \text{co-NP}$ , and widely believed to be in  $P$ . (The problem is trivially in  $P$  for DAGs.) The reason we include the PARITY problem in our survey is that it was the original motivation behind the development of some of the digraph width measures in the past decade (e.g., entanglement or DAG-width). The current standing of this problem with respect to digraph width measures is summarized in Theorem 4.22.

Here we give only an informal description of the game, and refer the reader to [GTW02] for a formal definition. Parity game is an infinite two-player game played on a digraph  $G = (V, E)$ , vertices of which are labeled by natural numbers. The players are called Odd and Even, and each vertex is “owned” by exactly one of the players. The play proceeds as follows. At the beginning, a token is placed on a special vertex of the graph (called the initial vertex). The game is then played in rounds. Let  $v$  be the current location of the token. Then the player who ‘owns’  $v$  moves the token along some arc  $(v, w) \in E$  to  $w$  (we require that there must be at least one such arc for each vertex). The player who owns  $w$  plays in the next round, in exactly the same way. This results in an infinite sequence of vertices, and we look at the sequence of numbers assigned to these vertices. The player Odd wins iff the least number in this sequence is odd, otherwise the player Even wins.

**Theorem 4.22.** *The PARITY problem is in XP for digraphs of*

- a) bounded tree-width ([Obd03]),*
- b) bounded entanglement (Berwanger and Grädel [BG05]),*
- c) bounded clique-width ([Obd07]),*
- d) bounded Kelly-width (Hunter and Kreutzer [HK08]),*
- e) bounded DAG-width (Berwanger et al. [BDH<sup>+</sup>12]).*

## 5. Conclusion

The main contribution of our paper is to give a thorough overview of the parameterized complexity status of many common digraph problems under different width measures as the parameters. The many existing and published results in this area, and the several new ones from the paper, are summarized in Table 2, completing it into an exhaustive reference guide to the field. Note that, regarding the positive entries in Table 2, there is no issue of whether a “decomposition” certifying low width is given along with the input or not; it is either not directly used by an algorithm (often the case with K-width and DAG-depth), or a suitable certificate can be itself computed in XP (parameterized by DAG-width [BDH<sup>+</sup>12] and cycle-rank) or in FPT (DFVS-number [CLL<sup>+</sup>08] and clique-width [HO08, KR13]).

Besides surveying the existing width measures and results, we introduce two new very restrictive measures (K-width and DAG-depth), which further confirm the presented overall picture that traditional structural measures do not fare well in parameterized algorithmics. For a closer explanation of the last sentence, notice that the positive entries in the first five columns of Table 2 are mostly incidental: These are concentrated in the rows of the HAM, *c*-PATH, and *c*-MINLOB problems, which are all three closely related to searching paths in a digraph and hence also to the considered structural measures. Otherwise, the only FPT entries in the first three columns of the table are the two remarkable ones of DFVS and KERNEL when parameterized by the DFVS-number itself. Further nontrivial polynomial entries lie in the column of DAGs, but that does not seem to extend to a useful parametrization.

The new K-width and DAG-depth are indeed very restrictive digraph measures, always at least as high as DAG-width and often much higher (see Table 1). From their respective definitions it seems that problems that stay NP-hard for digraphs with *constant* K-width and constant DAG-depth are likely far away from being fixed-parameter tractable for any similar width parameter of digraphs, and we have shown that various problems from several different areas are of this kind. This can be seen as a strong indication that DAG-width and related structural measures (which are often connected to variants of the cops-and-robber search game, Section 3.2) are not yet the right parameters for dealing with usual digraph problems. (Considering the DFVS number as a width parameter does not seem to help either.) One reason might be that cops “give”

Table 2: A summary of old and new (in boldface) parameterized complexity results on selected digraph width measures. ‡-marked entries indicate unknown lower complexity bound.

<b>Parameter</b>	K-width & DAG-depth	DAG-width & Cycle rank	DFVS-num.	DAGs	Clique-width
HAM (§4.2)	<b>FPT</b>	XP <sup>a</sup> / W[2]-h. <sup>b</sup>	XP <sup>a</sup> ‡	P	XP <sup>c</sup> / W[1]-h. <sup>d</sup>
<i>c</i> -PATH (§4.3)	<b>FPT</b>	XP <sup>a</sup> ‡	XP <sup>a</sup> ‡	P <sup>a</sup>	<b>FPT</b>
<i>k</i> -PATH (§4.3)	<b>para-NPC</b>	NPC <sup>e</sup>	NPC <sup>e</sup>	NPC <sup>e</sup>	para-NPC <sup>f</sup>
<i>c</i> -MINLOB (§4.4)	<b>FPT</b>	XP <sup>g</sup> / W[2]-h. <sup>b</sup>	XP <sup>g</sup> ‡	P <sup>h</sup>	XP <sup>c</sup> / W[1]-h. <sup>d</sup>
MINLOB (§4.4)	<b>para-NPC</b>	para-NPC <sup>g</sup>	<b>para-NPC</b>	P <sup>h</sup>	<b>XP<sup>j</sup> / W[1]-h.<sup>d</sup></b>
DiDS & DiST & MAXLOB (§4.5)	<b>para-NPC</b>	<b>NPC</b>	<b>NPC</b>	<b>NPC</b>	<b>FPT</b>
MAXDICUT (§4.6)	para-NPC <sup>b</sup>	NPC <sup>b</sup>	NPC <sup>b</sup>	NPC <sup>b</sup>	XP <sup>c</sup> / W[1]-h. <sup>k</sup>
<i>c</i> -OCN (§4.7)	<b>para-NPC</b>	NPC <sup>l</sup>	NPC <sup>l</sup>	NPC <sup>l</sup>	<b>FPT</b>
DFVS (§4.8)	<i>open</i>	para-NPC <sup>m</sup>	FPT <sup>n</sup>	P	<b>FPT</b>
KERNEL (§4.8)	para-NPC <sup>p</sup>	para-NPC <sup>m,p</sup>	<b>FPT</b>	P	<b>FPT</b>
$\phi$ -MSO <sub>1</sub> MC (§4.1)	<b>para-NPH</b>	NPH	NPH	NPH	FPT <sup>q</sup>
PARITY (§4.9)	XP <sup>r</sup> ‡	XP <sup>r</sup> ‡	XP <sup>r</sup> ‡	P	XP <sup>s</sup> ‡

<sup>a</sup>[JRST01] <sup>b</sup>[LKM11] <sup>c</sup>[GHO13] <sup>d</sup>[FGLS10b] <sup>e</sup>[EIS76] <sup>f</sup>[GW06] <sup>g</sup>[DGK09] <sup>h</sup>[GRK09]  
<sup>j</sup>[GHO11] <sup>k</sup>[FGLS10a] <sup>l</sup>[CD06] <sup>m</sup>[KO11] <sup>n</sup>[CLL<sup>+</sup>08] <sup>p</sup>[vL76] <sup>q</sup>[CMR00] <sup>r</sup>[BDH<sup>+</sup>12] <sup>s</sup>[Obd07]

good graph separators in the undirected case, but that does not work any more for digraphs. We refer to [GHK<sup>+</sup>10, GHL<sup>+</sup>12] for further investigation.

On the other hand, the last column of clique-width exposes a very different behavior: Not only that many tractable (FPT) cases follow directly from one metaresult – Theorem 4.2 about MSO<sub>1</sub> model checking, but there exist separate nontrivial XP algorithms for the other cases (except *k*-PATH), too. Among them, especially those for PARITY [Obd07] and MINLOB [GHO11] are remarkably involved, and show the algorithmic potential of the (mutually equivalent) clique-width and bi-rank-width measures on digraphs.

We conclude the paper with three noteworthy questions related to the summary:

- (Theorem 4.8) What is the complexity status of the *k*-PATH problem on digraphs of DAG-depth exactly 3?
- (Theorem 4.20) What is the parameterized complexity status of the DFVS problem with respect to DAG-depth and K-width?
- (Proposition 4.19) What is the parameterized complexity status of the OCN problem with respect to clique-width?



**Acknowledgements.** This work has been supported by a Czech–German bilateral grant of GAČR and DFG (201/09/J021 and RO 927/9). Currently, P. Hliněný and J. Obdržálek are supported by the research centre CE-ITI – Czech Science foundation project P202/12/G061, and R. Galian is supported by ERC (COMPLEX REASON, 239962).

## References

- [Adl07] I. Adler. Directed tree-width examples. *J. Comb. Theory Ser. B*, 97(5), 2007.
- [Bar06] J. Barát. Directed path-width and monotonicity in digraph searching. *Graphs and Combin.*, 22(2):161–172, 2006.
- [BDH<sup>+</sup>12] D. Berwanger, A. Dawar, P. Hunter, S. Kreutzer, and J. Obdržálek. The DAG-width of directed graphs. *J. Comb. Theory, Ser. B*, 102(4):900–923, 2012.
- [BG05] D. Berwanger and E. Grädel. Entanglement – A measure for the complexity of directed graphs with applications to logic and games. In *LPAR’04*, volume 3452 of *LNCS*, pages 209–223. Springer, 2005.
- [BJK09] J. Bang-Jensen and M. Kriesell. Disjoint directed and undirected paths and cycles in digraphs. *Theor. Comput. Sci.*, 410(47-49):5138–5144, 2009.
- [BK08] H. Bodlaender and A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, 2008.
- [Bod93] H. Bodlaender. A tourist guide through treewidth. *Acta Cybernet.*, 11:1–21, 1993.
- [Bod98] H. Bodlaender. A partial  $k$ -arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209:1–45, 1998.
- [CD06] J.-F. Culus and M. Demange. Oriented coloring: Complexity and approximation. In *SOFSEM’06*, volume 3831 of *LNCS*, pages 226–236. Springer, 2006.
- [CKL<sup>+</sup>09] J. Chen, J. Kneis, S. Lu, D. Mölle, S. Richter, P. Rossmanith, S. Sze, and F. Zhang. Randomized divide-and-conquer: Improved path, matching, and packing algorithms. *SIAM J. Comput.*, 38(6):2526–2547, 2009.
- [CLL<sup>+</sup>08] J. Chen, Y. Liu, S. Lu, B. O’Sullivan, and I. Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM*, 55(5), 2008.

- [CMR00] B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000.
- [CO00] B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Appl. Math.*, 101(1-3):77–114, 2000.
- [Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158, 1971.
- [DF99] R. Downey and M. Fellows. *Parameterized complexity*. Monographs in Computer Science. Springer, 1999.
- [DGK09] P. Dankelmann, G. Gutin, and E. Kim. On complexity of minimum leaf out-branching problem. *Discrete Appl. Math.*, 157(13):3000–3004, 2009.
- [Die05] R. Diestel. *Graph Theory*, volume 173 of *Graduate texts in mathematics*. Springer, New York, 2005.
- [EF99] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1999.
- [Egg63] L. Eggan. Transition graphs and the star-height of regular events. *Michigan Math. J.*, 10(4):385–397, 1963.
- [EIS76] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM J. Comput.*, 5(4):691–703, 1976.
- [FG06] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.
- [FGLS10a] F. Fomin, P. Golovach, D. Lokshtanov, and S. Saurab. Algorithmic lower bounds for problems parameterized by clique-width. In *SODA'10*, pages 493–502. SIAM, 2010.
- [FGLS10b] F. Fomin, P. Golovach, D. Lokshtanov, and S. Saurab. Intractability of clique-width parameterizations. *SIAM J. Comput.*, 39(5):1941–1956, 2010.
- [FHW80] S. Fortune, J. E. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoret. Comput. Sci.*, 10:111–121, 1980.
- [GH10] R. Ganian and P. Hliněný. On parse trees and Myhill–Nerode–type tools for handling graphs of bounded rank-width. *Discrete Appl. Math.*, 158:851–867, 2010.
- [GHK<sup>+</sup>10] R. Ganian, P. Hliněný, J. Kneis, D. Meister, J. Obdržálek, P. Rossmanith, and S. Sikdar. Are there any good digraph width measures? In *IPEC'10*, volume 6478 of *LNCS*, pages 135–146. Springer, 2010.

- [GHL<sup>+</sup>12] R. Ganian, P. Hliněný, A. Langer, J. Obdržálek, P. Rossmanith, and S. Sikdar. Lower bounds on the complexity of MSO1 model-checking. In *STACS'12*, volume 14 of *LIPICs*, pages 326–337. Dagstuhl Publishing, 2012.
- [GHO11] R. Ganian, P. Hliněný, and J. Obdržálek. Clique-width: When hard does not mean impossible. In *STACS'11*, volume 9 of *LIPICs*, pages 404–415. Dagstuhl Publishing, 2011.
- [GHO13] R. Ganian, P. Hliněný, and J. Obdržálek. Unified approach to polynomial algorithms on graphs of bounded (bi-)bank-width. *European J. Combin.*, 34(3):680–701, 2013.
- [GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, San Francisco, 1979.
- [GRK09] G. Gutin, I. Razgon, and E. J. Kim. Minimum leaf out-branching and related problems. *Theoret. Comput. Sci.*, 410(45):4571–4579, 2009.
- [Gru08] H. Gruber. Digraph complexity measures and applications in formal language theory. In *MEMICS'08*, pages 60–67, 2008.
- [GT11] A. Giannopoulou and D. Thilikos. A min-max theorem for LIFO-search. *Electron. Notes in Discrete Math.*, 38:395 – 400, 2011.
- [GTW02] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games*, volume 2500 of *LNCS*. Springer, 2002.
- [GW06] F. Gurski and E. Wanke. Vertex disjoint paths on clique-width bounded graphs. *Theoret. Comput. Sci.*, 359(1-3):188–199, 2006.
- [HK08] P. Hunter and S. Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. *Theoret. Comput. Sci.*, 399(3):206–219, 2008.
- [HO06] P. Hliněný and J. Obdržálek. Escape-width: Measuring ”width” of digraphs. Presented at Sixth Czech-Slovak International Symposium on Combinatorics, Graph Theory, Algorithms and Applications, 2006.
- [HO08] P. Hliněný and S. Oum. Finding branch-decomposition and rank-decomposition. *SIAM J. Comput.*, 38:1012–1032, 2008.
- [HOSG08] P. Hliněný, S. Oum, D. Seese, and G. Gottlob. Width parameters beyond tree-width and their applications. *The Computer Journal*, 51(3):326–362, 2008.
- [HRW92] F. Hwang, D. Richards, and P. Winter. *The Steiner Tree Problem*. Ann. Discrete Math. Noth-Holland, 1992.

- [JRST01] T. Johnson, N. Robertson, P. D. Seymour, and R. Thomas. Directed tree-width. *J. Combin. Theory Ser. B*, 82(1):138–154, 2001.
- [KLR08] J. Kneis, A. Langer, and P. Rossmanith. A new algorithm for finding trees with many leaves. In *ISAAC'08*, volume 5369 of *LNCS*, pages 270–281. Springer, 2008.
- [KLR11] J. Kneis, A. Langer, and P. Rossmanith. Courcelle’s theorem - a game-theoretic approach. *Discrete Optim.*, 8(4):568–594, 2011.
- [KM04] W. Klostermeyer and G. MacGillivray. Homomorphisms and oriented colorings of equivalence classes of oriented graphs. *Discrete Mathematics*, 274:161–172, 2004.
- [KO11] S. Kreutzer and S. Ordyniak. Digraph decompositions and monotonicity in digraph searching. *Theor. Comput. Sci.*, 412(35):4688–4703, 2011.
- [KR13] M. Kanté and M. Rao. The rank-width of edge-coloured graphs. *Theory Comput. Syst.*, 52(4):599–644, 2013.
- [Kro67] M. R. Krom. The decision problem for a class of first-order formulas in which all disjunctions are binary. *Mathematical Logic Quarterly*, 13(1-2):15–20, 1967.
- [Lev73] Leonid Levin. Universal sequential search problems. In *Problems of Information Transmission*, volume 9, pages 265–266, 1973.
- [LKM11] M. Lampis, G. Kaouri, and V. Mitsou. On the algorithmic effectiveness of digraph decompositions and complexity measures. *Discrete Optim.*, 8(1):129–138, 2011.
- [NdM06] J. Nešetřil and P. Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *European J. Combin.*, 27(6):1024–1041, 2006.
- [Nie10] R. Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In *STACS'10*, volume 5 of *LIPICs*, pages 17–32. Schloss Dagstuhl, 2010.
- [Obd03] J. Obdržálek. Fast mu-calculus model checking when tree-width is bounded. In *CAV 2003*, volume 2725 of *LNCS*, pages 80–92. Springer, 2003.
- [Obd07] J. Obdržálek. Clique-width and parity games. In *CSL'07*, volume 4646 of *LNCS*, pages 54–68. Springer, 2007.
- [Rab08] R. Rabinovich. Complexity measures for directed graphs. Diploma thesis, RWTH Aachen, 2008.

- [RS86] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.
- [RS91] N. Robertson and P. D. Seymour. Graph minors. X. Obstructions to tree-decomposition. *J. Combin. Theory Ser. B*, 52(2):153–190, 1991.
- [Saf05] M. Safari. D-width: A more natural measure for directed tree-width. In *MFCS'05*, volume 3618 of *LNCS*, pages 745–756. Springer, 2005.
- [Sli10] A. Slivkins. Parameterized tractability of edge-disjoint paths on directed acyclic graphs. *SIAM J. Discrete Math.*, 24(1):146–157, 2010.
- [ST93] P. D. Seymour and R. Thomas. Graph searching and a min-max theorem for tree-width. *J. Combin. Theory Ser. B*, 58(1):22–33, 1993.
- [vL76] J. van Leeuwen. Having a Grundy-numbering is NP-complete. Technical Report 207, The Pennsylvania State University, September 1976.