On width measures and topological problems on semi-complete digraphs

Fedor V. Fomin [*]   Michał Pilipczuk [*]

Abstract

The topological theory for semi-complete digraphs, pioneered by Chudnovsky, Fradkin, Kim, Scott, and Seymour [11, 12, 13, 30, 31, 42], concentrates on the interplay between the most important width measures — cutwidth and pathwidth — and containment relations like topological/minor containment or immersion. We propose a new approach to this theory that is based on outdegree orderings and new families of obstacles for cutwidth and pathwidth. Using the new approach we are able to reprove the most important known results in a unified and simplified manner, as well as provide multiple improvements. Notably, we obtain a number of efficient approximation and fixed-parameter tractable algorithms for computing width measures of semi-complete digraphs, as well as fast fixed-parameter tractable algorithms for testing containment relations in the semi-complete setting. As a direct corollary of our work, we also derive explicit and essentially tight bounds on duality relations between width parameters and containment orderings in semi-complete digraphs.

[*]Department of Informatics, University of Bergen, Bergen, Norway, {fomin,michal.pilipczuk}@ii.uib.no. Preliminary results of this work were presented at SODA 2013, STACS 2013, and ESA 2013 [27, 43, 28]. An exposition of these results can be also found in the PhD thesis of the second author [44]. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 267959.

Contents

1 Introduction

1.1 The theory of graph minors for digraphs

The *Graph Minors* series of Robertson and Seymour not only resolved the Wagner's Conjecture [48], but also provided a number of algorithmic tools for investigating topological structure of graphs. It is a very natural and important question, whether techniques and results of Graph Minors can be applied in the world of *directed* graphs or digraphs. In spite of many attempts, we are still very far from the right answer. Even to capture a good analogue for treewidth in digraphs is a non-trivial task and several notions like directed treewidth [38], DAG-width [7] or Kelly-width [37] can be found in the literature. However, none of them shares all the "nice" properties of undirected treewidth. In fact this claim can be formalized and proved; Ganian et al. [33] argued that "*any reasonable algorithmically useful and structurally nice digraph measure cannot be substantially different from the treewidth of the underlying undirected graph*".

The notion of a graph minor is crucial in defining obstructions to small treewidth in an undirected graph. There are several ways to generalize this definition to digraphs and, as in case of treewidth, it is unclear which of them is the most natural. One approach is to consider topological embeddings or immersions. An undirected graph H is a *topological subgraph* (or *topological minor*) of an undirected graph G if a subdivision of H is a subgraph of G. In other words, graph H can be embedded into graph G in such a way that vertices of H are mapped to pairwise different vertices of G, and edges of H are mapped to vertex-disjoint paths in G. An *immersion* of a graph H into a graph G is defined like a topological embedding, except that edges of H correspond to edge-disjoint paths in G. Both these notions can be naturally translated to directed graphs by replacing paths with directed paths.

It were long-standing open questions whether deciding if an undirected graph H can be topologically embedded (immersed) into G is fixed-parameter tractable[1] when parameterized by the size of H. Both questions were answered positively only very recently by Grohe, Kawarabayashi, Marx, and Wollan [35]. Unfortunately, the work of Grohe et al. cannot be extended to directed graphs. By the classical result of Fortune, Hopcroft, and Wyllie [29] the problem of testing whether a given digraph G contains H as a (directed) topological subgraph is NP-complete even for very simple digraphs H of constant size. Similar results can be easily obtained for immersions. In fact, what Fortune et al. [29] showed is that the VERTEX (EDGE) DISJOINT PATHS problems are NP-complete on general digraphs even for $k = 2$, and the hardness of topological containment and immersion testing are simple corollaries of this fact.

Therefore, VERTEX (EDGE) DISJOINT PATHS were studied intensively on different classes of directed graphs. For example, if we constrain the input digraphs to be acyclic, then both variants still remain NP-complete when k is part of the input [22], but are polynomial-time solvable for every constant number of terminal pairs k [29], which was not the case in the general setting [29]. Slivkins [49] has shown that both problems are in fact $W[1]$-hard when parameterized by k and hence probably not FPT, which completes the picture of their parameterized complexity in this restricted case.

[1]Recall that a parameterized computational problem is called *fixed-parameter tractable* (FPT) if it can be solved in time $f(k) \cdot n^c$, where k is the parameter, n is the total input size, $f(\cdot)$ is some computable function and c is a constant independent of the parameter. Similarly, a problem is *in XP* if it admits an algorithm with running time $\mathcal{O}(n^{f(k)})$ for some computable function $f(\cdot)$. For more information on parameterized complexity we refer to the books of Downey and Fellows [19] and of Flum and Grohe [25].

1.2 The containment theory for tournaments

Tournaments form an interesting and mathematically rich subclass of digraphs. Formally, a simple digraph T is a tournament if for every pair of vertices v, w, *exactly* one of arcs (v, w) or (w, v) is present in T. We also consider a superclass of tournaments, called semi-complete digraphs, where we require *at least* one of arcs (v, w) or (w, v) to be present in T, thus allowing both of them to be present at the same time. Many algorithmic problems were studied on tournaments, with notable examples of problems strongly related to this work: Vertex (Edge) Disjoint Paths and Feedback Arc (Vertex) Set problems. We refer to the book of Bang-Jensen and Gutin [5] for a more thorough introduction to algorithms for digraphs, and in particular for tournaments.

The work on topological problems in semi-complete digraphs began perhaps with the work of Bang-Jensen and Thomassen [4, 6], who showed that in spite of the fact that the Vertex (Edge) Disjoint Paths problems remain NP-complete on tournaments when k is a part of the input, they are solvable in polynomial time for the case $k = 2$ even on semi-complete digraphs. This line of research was recently continued by Chudnovsky, Fradkin, Kim, Scott, and Seymour [11, 12, 13, 30, 31, 42] (CFKSS, for short), who drastically advanced the study of minor-related problems in semi-complete digraphs by building an elegant containment theory for this class. The central notions of the theory are two width measures of digraphs: *cutwidth* and *pathwidth*. The first one is based on vertex orderings and resembles classical cutwidth in the undirected setting [50], with the exception that only arcs directed forward in the ordering contribute to the cut function. The second one is a similar generalization of the undirected pathwidth.

Chudnovsky, Fradkin, and Seymour [11] proved a structural theorem that provides a set of obstacles for admitting an ordering of small (cut)width; a similar theorem for pathwidth was proven by Fradkin and Seymour [31]. A large enough obstacle for cutwidth admits every fixed-size digraph as an immersion, and the corresponding is true also for pathwidth and topological containment. Thus, these results can be reinterpreted as weak duality of the width measures and the containment relations in semi-complete digraphs. Basing on the first result, Chudnovsky and Seymour [13] were able to show that immersion is a well-quasi-ordering on the class of semi-complete digraphs. Indeed, following the same line of reasoning as in the Graph Minors project, it is sufficient to prove the claim for the class of semi-complete digraphs that exclude some fixed semi-complete digraph H_0 as an immersion. Using the structural theorem we infer that such digraphs have cutwidth bounded by a constant. For graphs of constant cutwidth, however, the well-quasi-ordering claim can be proven using a more direct approach via Higman's lemma.

Unfortunately, the same reasoning breaks for topological containment, since it is already not true that topological containment is a well-quasi-ordering of semi-complete digraphs of constant pathwidth. However, Kim and Seymour [42] have recently introduced a slightly different notion of a *minor* order, which indeed is a well-quasi-ordering of semi-complete digraphs.

As far as the algorithmic aspects of the work of CFKSS are concerned, the original proofs of the structural theorems can be turned into approximation algorithms, which given a semi-complete digraph T and an integer k find a decomposition of T of width $\mathcal{O}(k^2)$, or provide an obstacle for admitting a decomposition of width at most k. For cutwidth the running time on an n-vertex semi-complete digraph is $\mathcal{O}(n^3)$, but for pathwidth it is $\mathcal{O}(n^{f(k)})$ for some function f; this excludes usage of this approximation as a subroutine in any FPT algorithm, e.g. for topological containment testing.

As Chudnovsky, Fradkin, and Seymour [11] observed, the existence of an FPT approximation algorithm for cutwidth allows us to design FPT algorithms for deciding whether a given digraph H can be immersed into a semi-complete digraph T. Consider the following WIN/WIN approach. We run the approximation algorithm for cutwidth for some parameter that is a (large) function of $|H|$. In case an ordering of width bounded by a function of $|H|$ is returned, we can employ a dynamic programming routine on this decomposition that solves the problem in FPT time. Otherwise, the approximation algorithm provided us with a large combinatorial obstacle into which every digraph of size at most $|H|$ can be embedded. Therefore, we can safely provide a positive answer. Fradkin and Seymour [31] observe that the same approach can be applied to topological subgraph testing using their approximation algorithm for pathwidth instead. However, this approximation algorithm does not work in FPT time, so the obtained topological containment test is also not fixed-parameter tractable. Let us remark that the original dynamic programming routine for topological containment working on a path decomposition, presented by Fradkin and Seymour [31], was also not fixed-parameter tractable.

The approximation algorithms for cutwidth and for pathwidth either provide an obstacle into which every digraph of size at most ℓ can be embedded, or construct a decomposition of width $f(\ell)$ for some multiple-exponential function f (yet elementary). Therefore, the obtained algorithm for immersion testing also inherits this multiple-exponential dependence on the size of the digraph to be embedded, i.e., it works in time $f(|H|)\cdot n^3$ for some function f that is multiple-exponential, yet elementary. For topological containment this problem is even more serious, since we obtain multiple-exponential dependence on $|H|$ in the exponent of the polynomial factor, and not just in the multiplicative constant standing in front of it.

One of the motivations of the work of CFKSS was extending the work of Bang-Jensen and Thomassen on the Vertex (Edge) Disjoint Paths problems [6]. The new containment theory turned out to be capable of answering many questions, yet not all of them. Using the approximation algorithm for cutwidth, Fradkin and Seymour [30] designed an FPT algorithm for Edge Disjoint Paths working in time $f(k)\cdot n^5$ for some function f that is multiple-exponential, yet elementary. The algorithm uses again the WIN/WIN approach: having approximated cutwidth, we can either find an ordering of small width on which a dynamic program can be employed, or we find a large combinatorial obstacle. In the latter case, Chudnovsky, Fradkin, and Seymour are able to identify an irrelevant vertex in the obstacle, which can be safely removed without changing the existence of a solution. That is, they design an irrelevant vertex rule. For the Vertex Disjoint Paths problem, Chudnovsky, Scott, and Seymour [12] give an XP, i.e. of running time $\mathcal{O}(n^{f(k)})$, algorithm using a different approach. To the best of our knowledge, the question whether the Vertex Disjoint Paths problem admits an FPT algorithm on semi-complete digraphs is still open.

The Edge Disjoint Paths problem is a special case of the Rooted Immersion problem defined as follows: given a digraph H with prescribed pairwise different vertices $u_1, u_2, \ldots, u_h$, called *roots*, and a semi-complete digraph T also with pairwise different roots $v_1, v_2, \ldots, v_h$, we ask whether there exists an immersion of H in T that *preserves roots*, that is, maps each u_i to corresponding v_i. The Edge Disjoint Paths problem can be hence modeled as follows: we take H to be a digraph consisting of k independent arcs, and all the vertices of H are roots required to be mapped to respective endpoints of the paths that we seek for. In the same manner we may define Rooted Topological Containment problem that generalizes the Vertex Disjoint Paths problem. It appears that the FPT algorithm for Edge Disjoint Paths of Fradkin and Seymour [30] can be generalized to solve also the Rooted Infusion problem, which is a relaxation of Rooted Immersion where we do not require the images of vertices of H to be distinct. Note that Rooted Infusion also generalizes Edge Disjoint Paths in the same manner as Rooted Immersion does. As Fradkin and Seymour admit, they were not able to solve the Rooted Immersion problem in FPT time using their approach.

As far as computation of cutwidth and pathwidth exactly is concerned, by the well-quasi-ordering result of Chudnovsky and Seymour [13] we have that the class of semi-complete digraphs of cutwidth bounded by a constant is characterized by a finite set of forbidden immersions; the result of Kim and Seymour [42] proves that we can infer the same conclusion about pathwidth and minors. Having approximated the corresponding parameter in FPT or XP time, we can check if any of these forbidden structures is contained in a given semi-complete digraph using dynamic programming. This gives FPT and XP exact algorithms for computing cutwidth and pathwidth, respectively; however, they are both non-uniform — the algorithms depend on the set of forbidden structures which is unknown — and non-constructive — they provide just the value of the width measure, and not the optimal decomposition. Also, we have virtually no control over the dependence of the running time on the target width value.

1.3 Our results and techniques

In this paper, we address a number of open questions about the fixed-parameter tractability of topological problems in semi-complete digraphs that arose from the containment theory of CFKSS. Efficient computation of the two width measures of semi-complete digraphs are crucial for our algorithms for containment problems. Figure 1 contains a summary of our results.

Problem	Previous results	This work
Cutwidth approximation	$\mathcal{O}(n^3)$ time, width $\mathcal{O}(k^2)$ [11]	$\mathcal{O}(n^2)$ time, width $\mathcal{O}(k^2)$ (Thm 5.2)
Cutwidth exact	$f(k) \cdot n^3$ time, non-uniform, non-constructive [11, 13]	$2^{\mathcal{O}(\sqrt{k \log k})} \cdot n^2$ time (Thm 5.22)
Pathwidth approximation	$\mathcal{O}(n^{\mathcal{O}(k)})$ time, width $\mathcal{O}(k^2)$ [31]	$\mathcal{O}(kn^2)$ time, width $6k$ (Thm 4.12)
Pathwidth exact	$n^{m(k)}$ time, non-uniform, non-constructive [42]	$2^{\mathcal{O}(k \log k)} \cdot n^2$ time (Thm 4.13)
Topological containment	$\mathcal{O}(n^{m(\|H\|)})$ time [31]	$2^{\mathcal{O}(\|H\| \log \|H\|)} \cdot n^2$ time (Thm 6.5)
Immersion	$f(\|H\|) \cdot n^3$ time [11]	$2^{\mathcal{O}(\|H\|^2 \log \|H\|)} \cdot n^2$ time (Thm 6.6)
Minor	$\mathcal{O}(n^{m(\|H\|)})$ time	$2^{\mathcal{O}(\|H\| \log \|H\|)} \cdot n^2$ time (Thm 6.7)
Rooted Immersion	-	$f(\|H\|) \cdot n^3$ time (Thm 9.9)
Edge Disjoint Paths	$f(k) \cdot n^5$ time [30]	$f(k) \cdot n^3$ time (Thm 9.9)

Figure 1: Comparison of previously known algorithms and the results of this paper. The algorithms for cutwidth and pathwidth take as input a semi-complete digraph on n vertices and an integer k. The approximation algorithms can output a decomposition of larger width (guarantees are in the corresponding cells) or conclude that a decomposition of width at most k does not exist. The exact algorithms either construct a decomposition of width at most k or conclude that this is impossible. We remark that the XP algorithm for testing the minor relation has not been stated explicitly in any of the previous works, but it follows from them easily.

Width measures. We give several efficient algorithms for computing the cutwidth and the pathwidth of a semi-complete digraph. Our approach to compute these parameters is quite different from the approach used in [11, 13, 31]. The crucial observation of our approach can be intuitively formulated as follows: every ordering of vertices with respect to non-decreasing outdegrees is a good approximation of the order in which the vertices appear in some path decomposition close to optimal. In fact, for cutwidth this is true even in the formal sense. We prove that any outdegree ordering of vertices of a semi-complete digraph T has width at most $\mathcal{O}(\mathbf{ctw}(T)^2)$, hence we have a trivial approximation algorithm that sorts the vertices with respect to their outdegrees. The approach based on degree orderings brings us not only to better algorithms, it allows to reprove the structural theorems for both parameters in a unified and simplified way.

In the case of pathwidth, which is of our main interest, the intuitive outdegree ordering argument can be formalized as follows: If a semi-complete digraph T contains $4k+2$ vertices whose outdegrees pairwise differ by at most k, then the pathwidth of T is more than k. We call this obstacle a *degree tangle*. Hence, any outdegree ordering of vertices of a given semi-complete digraph T of small pathwidth must be already quite spread: it does not contain larger clusters of vertices with similar outdegrees. This spread argument is crucial in all our reasonings.

Both the approximation and the exact algorithm for pathwidth use the concept of scanning through the outdegree ordering with a *window* — an interval in the ordering containing $4k$ vertices. By the outdegree spread argument, at each point we know that the vertices on the left side of the window have outdegrees smaller by more than k than the ones on the right side; otherwise we would have a too large degree tangle. For approximation, we construct the consecutive bags by greedily taking the window and augmenting this choice with a small coverage of arcs jumping over it. The big gap between outdegrees on the left and on the right side of the window ensures that the nonexistence of a small coverage is also an evidence for not admitting a path decomposition of small width. The obtained approximation ratio is 6. For the exact algorithm, we identify a set of $\mathcal{O}(k^2)$ vertices around the window, about which we can safely assume that the bag is contained in it. This provides us $2^{\mathcal{O}(k \log k)}$ candidates for each bag of the optimal path decomposition. Using the candidates, the optimal path decomposition can be then assembled using dynamic programming in FPT time.

The most technical part in the approximation and exact algorithms for pathwidth is the choice of vertices outside the window to cover the arcs jumping over it. It turns out that this problem can be expressed as trying to find a small vertex cover in an auxiliary bipartite graph. However, in order to obtain a feasible path decomposition we cannot choose the vertex cover arbitrarily — it must behave consistently as the window slides through the ordering. To this end, in the approximation algorithm we use a 2-approximation of the vertex cover based on the theory of matchings in bipartite graphs. In the exact algorithm we need more restrictions, as we seek a subset that contains *every* sensible choice of the bag. Therefore, we use an $\mathcal{O}(OPT)$-approximation of vertex cover based on the classical kernelization routine for the problem of Buss [8], which enables us to use stronger arguments to reason which vertices can be excluded from consideration.

We believe that the new obstacles for pathwidth are more useful from the algorithmic perspective than the ones introduced by Fradkin and Seymour [31]. In particular, we show that the ratio between the sizes of obstacles found by the algorithm and the minimum size of a digraph that cannot be embedded into them does not exceed some constant. In other words, we prove that there exists a linear relation between the pathwidth and the size of the smallest H that is not topologically contained in a semi-complete digraph. Hence, in all the containment tests we just need to run the pathwidth approximation with parameter $\mathcal{O}(|H|)$, and in the case of finding an obstacle we can provide a positive answer. This reduces the dependence of the running time of all the containment tests to single exponential in terms of the size of the tested subgraph, compared to multiple-exponential following from the previous work.

We also show that the cutwidth of a semi-complete digraph can be computed in subexponential parameterized time, more precisely in time $2^{\mathcal{O}(\sqrt{k \log k})} \cdot n^2$. To achieve this running time, we have adopted the technique of *k-cuts*, developed earlier in [26] in the context of clustering problems. The main idea is to relate potentially useful states of the dynamic program (i.e., the aforementioned k-cuts) to *partition numbers*: the partition number $p(k)$ is the number of different multisets of positive integers summing up to k. The subexponential asymptotics of partition numbers have been very well understood from the point of view of enumerative combinatorics [20, 36], and we can use the results obtained there directly in our setting in order to bound the number of states of the dynamic program. This brings us to an algorithm deciding if the cutwidth of a semi-complete graph is at most k in time $2^{\mathcal{O}(\sqrt{k \log k})} \cdot n^2$ time.

As a byproduct of the approach taken, we also obtain a new algorithm for Feedback Arc Set in semi-complete digraphs with running time $\mathcal{O}(2^{c\sqrt{k}} \cdot k^{\mathcal{O}(1)} \cdot n^2)$ for $c = \frac{2\pi}{\sqrt{3} \cdot \ln 2} \leq 5.24$. Feedback Arc Set in Tournaments (FAST, for short) was perhaps the first natural parameterized problem outside the framework of bidimensionality [16] shown to admit a subexponential parameterized algorithm. The first such algorithm, with running time $2^{\mathcal{O}(\sqrt{k} \log k)} n^{\mathcal{O}(1)}$, is due to Alon, Lokshtanov, and Saurabh [3]. This has been further improved by Feige [23] and by Karpinski and Schudy [40], who have independently shown two different algorithms with running time $2^{\mathcal{O}(\sqrt{k})} n^{\mathcal{O}(1)}$. Our new algorithm is simpler than the aforementioned algorithms of Feige [23] and of Karpinski and Schudy [40]. While the asymptotic running time is similar, the explicit constant in the exponent obtained using our approach is much smaller than the constants in the algorithms of Feige and of Karpinski and Schudy; however, optimizing these constants was not the purpose of these works.

It appears that our approach can be also applied to other layout problems in semi-complete digraphs. For example, we consider a natural adaptation of the Optimal Linear Arrangement problem [10, 17] to the semi-complete setting, and we can decide in time $2^{\mathcal{O}(k^{1/3} \cdot \sqrt{\log k})} \cdot n^2$ whether a semi-complete digraph admits an ordering of cost at most k.

Topological problems. Using a polynomial-time approximation algorithm for pathwidth, we are able to design an FPT algorithm for testing topological containment using the same general WIN/WIN approach. Note here that to obtain this result one needs to implement the dynamic programming routine working on a path decomposition in FPT time, while the original routine of Fradkin and Seymour [31] was not fixed-parameter tractable. This, however, turns out to be less challenging, and it also follows from known tools on model checking $\mathbf{MSO}_1$ on digraphs of bounded cliquewidth. We give an introduction to tools borrowed from logic in Section 2.4, and design an explicit dynamic programming routine in Section 10 in order to give precise guarantees on the time complexity of the topological containment test. Algorithms for immersion and minor testing can be designed in a similar manner.

As explained earlier, the algorithms for testing containment relations are based on the classic WIN/WIN approach, and hence one can in fact prove that excluding some fixed digraph as a topological subgraph/minor/immersion implies an upper bound on pathwidth/cutwidth of a semi-complete digraph. These properties were already discovered by Fradkin and Seymour [31] and by Chudnovsky, Fradkin, and Seymour [11], but the obtained upper bounds were multiple exponential in terms of the size of the excluded digraph, and not provided explicitly. Using our results we can provide explicit and very low upper bounds: linear for pathwidth and quadratic for cutwidth. It is easy to show that these upper bounds are asymptotically tight: the relation must be at least linear for pathwidth and at least quadratic for cutwidth.

Fixed-parameter tractability of immersion and minor testing opens possibilities for proving meta-theorems of a more general nature via the well-quasi-ordering results of Chudnovsky and Seymour [13], and of Kim and Seymour [42]. We give an example of such a meta-theorem in

Section 8.

Another problem that we were able to address using the approximation algorithm for pathwidth, was the Rooted Immersion problem, whose parameterized complexity was left open by Fradkin and Seymour [30]. Unlike Fradkin and Seymour, we have approached the problem via pathwidth instead of cutwidth, having observed beforehand that a dynamic programming routine testing immersion on a path decomposition of small width can be designed similarly to topological containment. Hence, after running the approximation algorithm for pathwidth, either we can apply the dynamic programming on the obtained decomposition, or we are left with an obstacle for pathwidth, called a *triple*, which is more powerful than the obstacles for cutwidth used by Fradkin and Seymour. Similarly to Fradkin and Seymour, we design an irrelevant vertex rule on a triple, that is, in polynomial time we identify a vertex that can be safely removed from the triple without changing answer to the problem. Then we restart the algorithm. Observe that the algorithm can make at most n iterations before solving the problem by applying dynamic programming, since there are only n vertices in the digraph. All in all, this gives an algorithm for Rooted Immersion working in $f(|H|)\cdot n^3$ time for some elementary function f. Note that this in particular reduces the polynomial factor of the running time of the algorithm for Edge Disjoint Paths from n^5 to n^3.

Outline. This paper is organized as follows. Section 2 is devoted to preliminaries. After explaining notation and basic facts about semi-complete digraphs, we give formal definitions of containment notions, of width measures, and prove a number of results relating them. In particular, we show how the width measures relate to each other, and which width measures are closed under which containment relations. Apart from cutwidth and pathwidth that are of main interest in this work, we introduce also cliquewidth. The reason is that relations between cutwidth, pathwidth and cliquewidth in semi-complete digraphs are crucial for Section 2.4, where we gather the tools borrowed from logic that will be used later on. In Section 3, we define several sets of obstacles for width measures in semi-complete graphs. These combinatorial tools are used in Section 4, where we present the approximation and exact algorithms for pathwidth. Section 5 is devoted to cutwidth. We first give a simple approximation algorithm for this width measure, which follows directly from the results of Section 3. Then we give an subexponential parameterized algorithm for computing cutwidth exactly, and we utilize the same ideas to design similar algorithms for Feedback Arc Set and Optimal Linear Arrangement in semi-complete digraphs.

We give algorithms for testing containment relations in Section 6. Section 7 is devoted to the results on duality of width measures and containment relations, which are derived by slightly adjusting our algorithmic results. Then, in Section 8 we give an example of a meta-theorem that can be proven using a combination of our algorithms and the well-quasi-ordering results of Chudnovsky and Seymour [13], and of Kim and Seymour [42]. In Section 9, we give an algorithm for the Rooted Immersion problem. Section 10 contains descriptions of explicit dynamic programming routines for topological containment and immersion working on a path decomposition, which are used in Sections 6 and 9. Even though general construction of such routines follows directly from the tools borrowed from logic, we need to construct them explicitely in order to give precise upper bounds on the running times of the obtained algorithms. Finally, in Section 11 we conclude with open problems.

2 Preliminaries

2.1 Folklore and simple facts

In this section we provide some simple facts about semi-complete digraphs and tournaments that will be used in this paper. We begin with some observations on the out- and indegrees in semi-complete digraphs. Let T be a semi-complete digraph. Note that for every $v \in V(T)$ we have that $d^+(v) + d^-(v) \geq |V(T)| - 1$, and that the equality holds for all $v \in V(T)$ if and only if T is a tournament.

Lemma 2.1. *Let T be a semi-complete digraph. Then the number of vertices of T with outdegrees at most d is at most $2d + 1$.*

Proof. Let A be the set of vertices of T with outdegrees at most d, and for the sake of contradiction assume that $|A| > 2d + 1$. Consider semi-complete digraph $T[A]$. By a simple degree-counting argument, in every semi-complete digraph S there is a vertex of outdegree at least $\frac{|V(S)|-1}{2}$. Hence, $T[A]$ contains a vertex with outdegree larger than d. As outdegrees in $T[A]$ are not smaller than in T, this is a contradiction with the definition of A. □

Lemma 2.2. *Let T be a semi-complete digraph and let x, y be vertices of T such that $d^+(x) > d^+(y) + \ell$. Then there exist at least ℓ vertices that are both outneighbors of x and inneighbors of y and, consequently, ℓ vertex-disjoint paths of length 2 from x to y.*

Proof. Let $\alpha = d^+(y)$. We have that $d^-(y) + d^+(x) \geq |V(T)| - 1 - \alpha + \alpha + \ell + 1 = |V(T)| + \ell$. Hence, by the pigeonhole principle there exist at least ℓ vertices of T that are both outneighbors of x and inneighbors of y. □

We now proceed to a slight generalization of a folklore fact that every strongly connected tournament has a Hamiltonian cycle. In fact, this observation holds also in the semi-complete setting, and we include its proof for the sake of completeness.

Lemma 2.3. *A semi-complete digraph T has a Hamiltonian cycle if and only if it is strongly connected.*

Proof. Necessary condition being trivial, we proceed to the proof that every strongly connected semi-complete digraph T has a Hamiltonian cycle. We proceed by induction on $|V(T)|$. The base cases when T has one or two vertices are trivial, so we proceed with the assumption that $|V(T)| > 2$.

Let v be any vertex of T and let $T' = T \setminus v$. Let $T_1, T_2, \ldots, T_p$ be the strongly connected components of T'. Note that since T' is semi-complete, the directed acyclic graph of its strongly connected components must be also semi-complete, hence it must be a transitive tournament. Without loss of generality let $T_1, T_2, \ldots, T_p$ be ordered as in the unique topological ordering of this transitive tournament, i.e., for every $v_1 \in V(T_i)$ and $v_2 \in V(T_j)$, where $i \neq j$, we have that $(v_1, v_2) \in E(T')$ if and only if $i < j$. Since $T_1, T_2, \ldots, T_p$ are strongly connected, by inductive hypothesis let $C_1, C_2, \ldots, C_p$ be Hamiltonian cycles in $T_1, T_2, \ldots, T_p$, respectively.

Observe that there must be some vertex $v' \in V(T_1)$ such that $(v, v') \in E(T)$, as otherwise $(V(T_1), \{v\} \cup \bigcup_{i=2}^{p} V(T_i))$ would be a partition of $V(T)$ such that all the arcs between the left side and the right side of the partition are directed from the left to the right; this would be a contradiction with T being strongly connected. A symmetric reasoning shows that there exists some vertex $v'' \in V(T_p)$ such that $(v'', v) \in E(T)$.

We now distinguish two cases. In the first case we assume that $p = 1$. Let $v_1, v_2, \ldots, v_{n-1}$ be the vertices of $V(T') = V(T_1)$ ordered as on cycle C_1, i.e., $(v_i, v_{i+1}) \in E(T')$ where index i

behaves cyclically. Without loss of generality assume that $v_1 = v''$. We claim that there are two vertices v_i, v_{i+1} (where again i behaves cyclically) such that $(v, v_{i+1}) \in E(T)$ and $(v_i, v) \in E(T)$. If every vertex v_i is a tail of an arc directed towards v, then this claim is trivial: we just take index i such that $v_{i+1} = v'$. Otherwise there are some vertices that are not tails of arcs directed towards v, and let $i+1$ be the smallest index of such a vertex. Note that by the assumption that $v_1 = v''$ we have that $i+1 > 1$. Since T is semi-complete, it follows that $(v, v_{i+1}) \in E(T)$. By the minimality of $i+1$ and the fact that $i+1 > 1$, it follows that $(v_i, v) \in E(T)$, which proves that vertices v_i, v_{i+1} have the claimed property. We now can construct a Hamiltonian cycle C for the whole digraph T by inserting v between v_i and v_{i+1} in C_1; note that here we use the fact that $|V(T)| > 2$ so that v_i and v_{i+1} are actually two different vertices.

Now assume that $p > 1$. We construct a Hamiltonian cycle C for the whole T by concatenating cycles $C_1, C_2, \ldots, C_p$. To construct C, take first v and then place v' followed by the whole cycle C_1 traversed from v' to the predecessor of v'. Then proceed to an arbitrarily chosen vertex of C_2 and traverse the whole cycle C_2 from this vertex up to its predecessor. Continue in this manner through the consecutive components, but when considering C_p, instead of choosing an arbitrary vertex to begin with, choose the successor of v'' on C_p so that after traversing C_p we arrive at v'' that is a tail of an arc directed to v. It is easy to observe that C constructed in this manner is indeed a Hamiltonian cycle: it follows from the fact that $(v, v'), (v'', v) \in E(T)$, and for every two consecutive strongly connected components T_i, T_{i+1}, there is an arc from every vertex of the first component to every vertex of the second component. □

2.2 Definitions of containment relations

In this section we introduce formally the containment notions that will be of our interest. We start with the *immersion* and *topological containment* relations, which are direct analogues of the classical undirected versions. Then we proceed to the notion of *minor*, for which one needs to carefully describe how the undirected notion is translated to the directed setting.

Let H, G be digraphs. We say that mapping η is a *model* of H in G, if the following conditions are satisfied:

- for every vertex $v \in V(H)$, $\eta(v)$ is a subset of $V(G)$;
- for every arc $(u, v) \in E(H)$, $\eta((u, v))$ is a directed path leading from some vertex of $\eta(u)$ to some vertex of $\eta(v)$.

By imposing further conditions on the model we obtain various containment notions for digraphs. If we require that η:

- maps vertices of H to pairwise different singletons of vertices of G,
- and the paths in $\eta(E(H))$ are internally vertex-disjoint,

then we obtain the notion of *topological containment*. In this case we say that η is an *expansion* of H in G, and we say that H is a *topological subgraph* of G. As the images of vertices are singleton sets, we may think that η simply maps vertices of H to vertices of G. If we relax the condition on paths in $\eta(E(H))$ from being internally vertex-disjoint to being arc-disjoint, we arrive at the notion of *immersion*; then η is an *immersion* of H into G. Clearly, every expansion is also an immersion, so if G topologically contains H, then it contains H as an immersion as well.

Sometimes in the literature this notion is called *weak immersion* to distinguish it from *strong immersion*, where each image of an arc is additionally required not to pass through images of vertices not being the endpoints of this arc. In this work we are interested only in (weak)

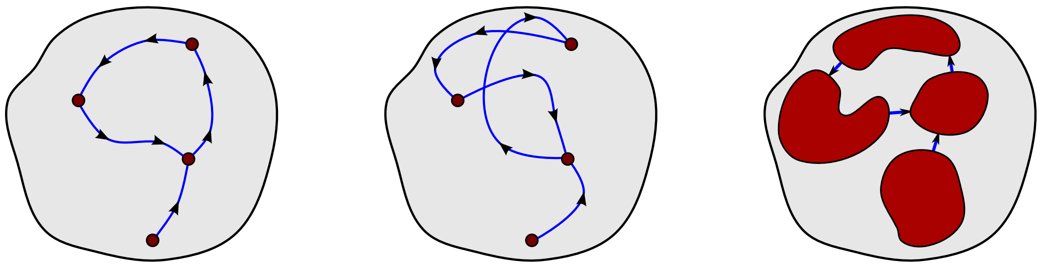
Figure 2: Expansion, immersion, and a minor model of the same 4-vertex digraph H in a larger digraph G.

immersions, as we find the notion of strong immersion not well-motivated enough to investigate all the technical details that arise when considering both definitions at the same time and discussing slight differences between them.

Finally, we proceed to the notion of a *minor*. For this, we require that η

- maps vertices of H to pairwise disjoint sets of vertices of G that moreover induce strongly connected subdigraphs,
- and maps arcs from $E(H)$ to arcs of $E(G)$ in such a manner that $\eta((u,v))$ is an arc from a vertex of $\eta(u)$ to a vertex of $\eta(v)$, for every $(u,v) \in E(H)$.

We then say that η is a *minor model* of H in G. In other words, we naturally translate the classical notion from the undirected graphs to the directed graphs by replacing the connectivity requirement with strong connectivity.

The notion of a minor of digraphs was introduced by Kim and Seymour [42], and we would like to remark that the original definition is more general as it handles also the case of digraphs with multiple arcs and loops. As in this work we are interested in simple digraphs only, we will work with this simplified definition, and we refer a curious reader to the work of Kim and Seymour [42] for more details on the general setting.

The introduced minor order is not directly stronger or weaker than the immersion or topological containment orders. In particular, contrary to the undirected setting, it is not true that if G contains H as a topological subgraph, then it contains H as a minor; as a counterexample take G being a triangle (a tournament being a directed cycle of length 3) and H being a complete digraph on 2 vertices (two vertices plus two arcs in both directions).

However, in the semi-complete setting the minor containment testing may be conveniently Turing-reduced to topological containment testing using the following lemma. In the following, we use the notion of *constrained topological containment*. We say that a digraph H is topologically contained in G *with constraints* $F \subseteq E(H)$, if all the images of arcs of F are of length 1, i.e., they are just single arcs in G.

Lemma 2.4. *There is an algorithm that, given a digraph H, in $2^{\mathcal{O}(|H|\log|H|)}$ time computes a family $\mathcal{F}_H$ of pairs (D,F) where D is a digraph and $F \subseteq E(D)$, with the following properties:*

(i) $|\mathcal{F}_H| \leq 2^{\mathcal{O}(|H|\log|H|)}$;

(ii) $|D| \leq 5|H|$ *for each* $(D,F) \in \mathcal{F}_H$;

(iii) for any semi-complete digraph T, H is a minor of T if and only if there is at least one pair $(D,F) \in \mathcal{F}_H$ such that D is topologically contained in T with constraints F.

Proof. We present first how the family $\mathcal{F}_H$ is constructed. For every vertex $u \in V(H)$, choose a number $p(u)$ between 1 and $d(u)$. Construct a directed cycle C_u of length $p(u)$ (in case $p(u) = 1$ take a single vertex without a loop), and let $u_1, u_2, \ldots, u_{p(u)}$ be vertices of C_u in this order. For every arc $(u, v) \in E(H)$, choose two integers $i, j = i((u, v)), j((u, v))$ such that $1 \leq i \leq p(u)$ and $1 \leq j \leq p(v)$, and add an arc (u_i, v_j). Family $\mathcal{F}_H$ consists of all the digraphs constructed in this manner, together with the sets of all the arcs not contained in cycles $C(u)$ as constraints.

Property (ii) follows directly from the construction, so let us argue that property (i) is satisfied. For every vertex u we have at most $d(u)$ choices of $p(u)$ and at most $d(u)^{d(u)}$ choices for integers $i((u, v))$ and $j((w, u))$ for $(u, v), (w, u) \in E(H)$. Hence, in total we have at most $\prod_{u \in V(H)} d(u)^{d(u)+1} = \prod_{u \in V(H)} 2^{\mathcal{O}(d(u) \log d(u))}$ choices. Since function $t \to t \log t$ is convex and $\sum_{u \in V(H)} d(u) = \mathcal{O}(|E(H)|)$, in total we will construct at most $2^{\mathcal{O}(|H| \log |H|)}$ digraphs. As each digraph from H is constructed in polynomial time, the running time of the algorithm also follows.

We are left with proving that property (iii) is satisfied as well. Assume first that we are given a semi-complete digraph T, and there is a pair $(D, F) \in \mathcal{F}_H$ such that T contains expansion η of some D, where the arcs of F are mapped to single arcs in T. Since cycles $C(u)$ in D are strongly connected, so do their images in η. Hence, to construct a minor model of H in G, we can simply map every vertex of $u \in V(H)$ to $V(\eta(C(u)))$. Existence of appropriate arcs modeling arcs of $E(H)$ follows from the fact that arcs of F are mapped to single arcs in T in η.

Assume now that G admits a minor model η of H. For every set $\eta(u)$ for $u \in V(H)$, construct a Hamiltonian cycle $C_0(u)$ in $T[\eta(u)]$ using Lemma 2.3. Then define a digraph D by taking T, and

- removing all the vertices not participating in any cycle $C_0(u)$,
- removing all the arcs not participating in any cycle $C_0(u)$ and not being images of arcs of H in η,
- and contracting all the paths of vertices with degrees 2 on cycles $C_0(u)$ to single arcs.

Observe that since at most $d(u)$ vertices of $C_0(u)$ are incident to images of arcs of H incident to u, then cycle $C_0(u)$ after contractions have length at most $d(u)$. Therefore, it can be easily seen that the obtained digraph D is enumerated when constructing family $\mathcal{F}_H$, and moreover it is enumerated together with the set of images of arcs of H in η as constraints. Construction of D ensures that D is topologically contained in G with exactly these constraints. $\square$

One of the motivations of work of Chudnovsky, Fradkin, Kim, Scott,and Seymour, was finding containment notions that are well-quasi orders on the class of semi-complete digraphs. It appears that both immersion and minor orders have this property. As far as topological containment is concerned, Kim and Seymour [42] observe that it does not form a well-quasi-ordering.

Theorem 2.5 ([13])**.** *The immersion ordering is a well-quasi-ordering of the class of semi-complete digraphs.*[2]

Theorem 2.6 ([42])**.** *The minor ordering is a well-quasi-ordering of the class of semi-complete digraphs.*

Finally, we say that $\mathbf{G} = (G; v_1, \ldots, v_h)$ is a *rooted* digraph if G is digraph and $v_1, \ldots, v_h$ are pairwise different vertices of $V(G)$. The notions of immersion and topological containment

[2]Chudnovsky and Seymour state the result for tournaments only, but the proof works actually also in the semi-complete setting; cf. [42].

can be naturally generalized to rooted digraphs. Immersion η is an immersion from a rooted digraph $\mathbf{H} = (H; u_1, \ldots, u_h)$ to a rooted digraph $\mathbf{G} = (G; v_1, \ldots, v_h)$ if additionally $\eta(u_i) = v_i$ for $i \in \{1, \ldots, h\}$, that is, the immersion preserves the roots. Such an immersion is called an $\mathbf{H}$-*immersion* or a *rooted immersion.* In the same manner we may define $\mathbf{H}$-*expansions* or *rooted expansions.*

2.3 Width parameters

In this section we introduce formally the width notions of digraphs that will be used in this paper: cutwidth, pathwidth, and cliquewidth. As far as cutwidth and pathwidth are of our prime interest, we introduce cliquewidth for the sake of introducing meta-tools concerning model checking Monadic Second-Order logic that will make some of our later arguments cleaner and more concise. We first explain each of the parameters separately, and then proceed to proving inequalities between them.

2.3.1 Cutwidth

The notion of cutwidth of digraphs resembles the classical definition in the undirected setting, with an exception that only arcs directed forwards in the ordering contribute to the cut function.

Definition 2.7. *Given a digraph $G = (V, E)$ and an ordering π of V, let $\pi[\alpha]$ be the first α vertices in the ordering π. The* width *of π is equal to $\max_{0 \leq \alpha \leq |V|} |E(\pi[\alpha], V \setminus \pi[\alpha])|$; the* cutwidth *of G, denoted $\mathbf{ctw}(G)$, is the minimum width among all orderings of V.*

Note that any transitive tournament T has cutwidth 0: we simply take the reversed topological ordering of T. It appears that cutwidth is closed under taking immersions, i.e., if H is an immersion of G then $\mathbf{ctw}(H) \leq \mathbf{ctw}(G)$.

Lemma 2.8. *Let H, G be digraphs and assume that H can be immersed into G. Then $\mathbf{ctw}(H) \leq \mathbf{ctw}(G)$.*

Proof. Let σ be an ordering of $V(G)$ of width $\mathbf{ctw}(G)$ and let η be immersion of H into G. Define ordering σ' of $V(H)$ by setting $u <_{\sigma'} v$ if and only if $\eta(u) <_\sigma \eta(v)$. We claim that σ' has width at most $\mathbf{ctw}(G)$. Indeed, take any prefix $\sigma'[t']$ for $0 \leq t' \leq |V(H)|$ and corresponding suffix $V(H) \setminus \sigma'[t']$. By the definition of σ' we can chose a number t, $0 \leq t \leq |V(G)|$ such that $\eta(\sigma'[t']) \subseteq \sigma[t]$ and $\eta(V(H) \setminus \sigma'[t']) \subseteq V(G) \setminus \sigma[t]$. Now consider any arc $(u, v) \in E(H)$ such that $u \in \sigma'[t']$ and $v \in V(H) \setminus \sigma'[t']$; we would like to prove that the number of such arcs is at most $\mathbf{ctw}(G)$. However, $\eta((u, v))$ is a directed path from $\eta(u) \in \sigma[t]$ to $\eta(v) \in V(G) \setminus \sigma[t]$. All these paths are edge-disjoint and contain at least one arc in $E(\sigma[t], V(G) \setminus \sigma[t])$. As the number of such arcs is at most $\mathbf{ctw}(G)$, the lemma follows. $\square$

2.3.2 Pathwidth

Similarly to cutwidth, also the notion of pathwidth is a direct translation of the definition in the undirected setting, again bearing in mind the intuition that only arcs directed forwards in the decomposition are contributing to the width.

Definition 2.9. *Given a digraph $G = (V, E)$, a sequence $W = (W_1, \ldots, W_r)$ of subsets of V is a* path decomposition *of G if the following conditions are satisfied:*

(i) $\bigcup_{1 \leq i \leq r} W_i = V$;

(ii) $W_i \cap W_k \subseteq W_j$ *for* $1 \leq i < j < k \leq r$;

(iii) $\forall\ (u,v) \in E$, *either* $u, v \in W_i$ *for some* i *or* $u \in W_i$, $v \in W_j$ *for some* $i > j$.

We call $W_1, \ldots, W_r$ *the* bags *of the path decomposition. The* width *of a path decomposition is equal to* $\max_{1 \leq i \leq r}(|W_i| - 1)$*; the* pathwidth *of* G*, denoted* $\mathbf{pw}(G)$*, is the minimum width among all path decompositions of* G*.*

We would like to remark that conditions (i) and (ii) are equivalent to saying that for every vertex v, the set of bags containing v form a nonempty interval in the path decomposition. By I_v^W we denote this interval in W, treated as an interval of indices; in the notation we drop the decomposition W whenever it is clear from the context. Then condition (iii) is equivalent to saying that if (u,v) is an arc, then it cannot occur that I_u ends in the decomposition W before I_v starts. We use this equivalent definition interchangeably with the original one.

Note that Definition 2.9 formally allows having duplicate bags in the decomposition, thus making it possible for a path decomposition to have unbounded length. However, of course we may remove any number of consecutive duplicate bags while constructing the decomposition. Hence, we will implicitly assume that in all the path decompositions any two consecutive bags are different. Then, for any two consecutive bags W_i, W_{i+1} there exists a vertex $v \in V(T)$ such that interval I_v either ends in i or begins in $i+1$. Since there are $2|V(T)|$ beginnings and ends of intervals I_v for $v \in V(T)$ in total, we infer that any path decomposition W of T which does not contain any two consecutive equal bags has length at most $2|V(T)| + 1$.

Let us note that any transitive tournament T has pathwidth 0: we can construct a decomposition W of width 0 by taking singletons of all the vertices and ordering them according to the reversed topological ordering of T. Again, it appears that pathwidth is closed both under taking minors and topological subgraphs, i.e., if H is a topological subgraph or a minor of G then $\mathbf{pw}(H) \leq \mathbf{pw}(G)$. The proof of the first observation can be found in [44] and the second observation is due to Kim and Seymour [42].

For the sake of constructing dynamic programming routines it is convenient to work with *nice* path decompositions. We say that a path decomposition $W = (W_1, \ldots, W_r)$ is *nice* if it has following two additional properties:

- $W_1 = W_r = \emptyset$;
- for every $i = 1, 2, \ldots, r-1$ we have that $W_{i+1} = W_i \cup \{v\}$ for some vertex $v \notin W_i$, or $W_{i+1} = W_i \setminus \{w\}$ for some vertex $w \in W_i$.

If $W_{i+1} = W_i \cup \{v\}$ then we say that in bag W_{i+1} we *introduce* vertex v, while if $W_{i+1} = W_i \setminus \{w\}$ then we say that in bag W_{i+1} we *forget* vertex w. Given any path decomposition W of width p, in $\mathcal{O}(p|V(T)|)$ time we can construct a nice path decomposition W' of the same width in a standard manner: we first introduce empty bags at the beginning and at the end, and then between any two consecutive bags W_i, W_{i+1} we insert a sequence of new bags by first forgetting the vertices of $W_i \setminus W_{i+1}$, and then introducing the vertices of $W_{i+1} \setminus W_i$.

The following lemma uses the concept of a nice path decomposition to prove existence of vertices of small out- and indegrees.

Lemma 2.10. *Let* T *be a semi-complete digraph of pathwidth at most* k*. Then* T *contains a vertex of outdegree at most* k*, and a vertex of indegree at most* k*.*

Proof. Let $W = (W_1, \ldots, W_r)$ be a nice path decomposition of T of width at most k, and let v_0 be the vertex that is forgotten first in this path decomposition, i.e., the index i of the bag where it is forgotten is minimum. By minimality of i and the definition of path decomposition, every vertex that is an outneighbor of v_0 needs to be contained in W_i. Since there is at most k vertices other than v_0 in W_i, it follows that the outdegree of v_0 is at most k. The proof for indegree is symmetric — we take v_0 to be the vertex that is introduced last. $\square$

Path decompositions can be viewed also from a different perspective: a path decomposition naturally corresponds to a monotonic sequence of separations. This viewpoint will be very useful when designing exact and approximation algorithms for pathwidth.

Definition 2.11. *A sequence of separations $((A_0, B_0), \ldots, (A_r, B_r))$ is called a* separation chain *if $(A_0, B_0) = (\emptyset, V(T))$, $(A_r, B_r) = (V(T), \emptyset)$ and $A_i \subseteq A_j, B_i \supseteq B_j$ for all $i \leq j$. The* width *of the separation chain is equal to* $\max_{1 \leq i \leq r} |A_i \cap B_{i-1}| - 1$.

Lemma 2.12. *The following holds.*

- *Let $W = (W_1, \ldots, W_r)$ be a path decomposition of a digraph T of width at most p. Then sequence $((A_0, B_0), \ldots, (A_r, B_r))$ defined as $(A_i, B_i) = (\bigcup_{j=1}^{i} W_j, \bigcup_{j=i+1}^{r} W_j)$ is a separation chain in T of width at most p.*
- *Let $((A_0, B_0), \ldots, (A_r, B_r))$ be a separation chain in a digraph T of width at most p. Then $W = (W_1, \ldots, W_r)$ defined by $W_i = A_i \cap B_{i-1}$ is a path decomposition of T of width at most p.*

Proof. For the first claim, it suffices to observe that (A_i, B_i) are indeed separations, as otherwise there would be an edge $(v, w) \in E(T)$ such that v can belong only to bags with indices at most i and w can belong only to bags with indices larger than i; this is a contradiction with property (iii) of path decomposition. The bound on width follows from the fact that $A_i \cap B_{i-1} = W_i$ by property (ii) of path decomposition.

For the second claim, observe that the bound on width follows from the definition of a separation chain. It remains to carefully check all the properties of a path decomposition. Property (i) follows from the fact that $A_0 = \emptyset$, $A_r = V(T)$ and $W_i \supseteq A_i \setminus A_{i-1}$ for all $1 \leq i \leq r$. Property (ii) follows from the fact that $A_i \subseteq A_j, B_i \supseteq B_j$ for all $i \leq j$: the interval in the decomposition containing any vertex v corresponds to the intersection of the prefix of the chain where v belongs to sets A_i, and the suffix where v belongs to sets B_i.

For property (iii), take any $(v, w) \in E(T)$. Let α be the largest index such that $v \in W_\alpha$ and β be the smallest index such that $w \in W_\beta$. It suffices to prove that $\alpha \geq \beta$. For the sake of contradiction assume that $\alpha < \beta$ and consider separation (A_α, B_α). By maximality of α it follows that $v \notin B_\alpha$; as $\beta > \alpha$ and β is minimal, we have also that $w \notin A_\alpha$. Then $v \in A_\alpha \setminus B_\alpha$ and $w \in B_\alpha \setminus A_\alpha$, which contradicts the fact that (A_α, B_α) is a separation. □

Note that the transformations in the first and in the second claim of Lemma 2.12 are inverse to each other and can be carried out in $\mathcal{O}(p|V(T)|)$ assuming that we store separations along with separators. Hence, instead of looking for a path decomposition of width p one may look for a separation chain of width at most p. Note also that if decomposition W does not contain a pair of consecutive equal bags, then the corresponding separation chain has only separations of order at most p.

Assume that $((A_0, B_0), \ldots, (A_r, B_r))$ is a separation chain corresponding to a nice path decomposition W in the sense of Lemma 2.12. Since $A_0 = \emptyset$, $A_r = V(G)$, and $|A_i|$ can change by at most 1 between two consecutive separations, for every ℓ, $0 \leq \ell \leq |V(G)|$, there is some separation (A_i, B_i) for which $|A_i| = \ell$ holds. Let $W[\ell]$ denote any such separation; note that the order of $W[\ell]$ is at most the width of W.

Before we proceed, let us state one simple, but very important fact about separations in semi-complete digraphs. Assume that T is a semi-complete digraph and let (A, B) be a separation of T. We know that $E(A \setminus B, B \setminus A) = \emptyset$, so $E(B \setminus A, A \setminus B) = (B \setminus A) \times (A \setminus B)$ because T is semi-complete. A simple application of this observation is the following. If W is a nice path decomposition of T, then by Lemma 2.12 every bag W_i is a separator separating the vertices that are not yet introduced from the vertices that are already forgotten. Therefore, there is no arc from a vertex that is forgotten to a vertex that is not yet introduced, but from every vertex not yet introduced there is an arc to every vertex that is already forgotten.

2.3.3 Cliquewidth

A *labeled digraph* is a pair (D, α), where D is a digraph and $\alpha : V(D) \to \{1, 2, \dots, k\}$ is a labeling function that associates with each vertex of D one of k different labels. We define three operations on labeled digraphs:

- **Disjoint union** $\oplus$ is defined as
$$(D_1, \alpha_1) \oplus (D_2, \alpha_2) = (D_1 \cup D_2, \alpha_1 \cup \alpha_2).$$
In other words, we take the disjoint union of digraphs D_1 and D_2, and define the labeling as the union of original labelings.
- **Join** $\eta_{i,j}(\cdot)$ for $i, j \in \{1, 2, \dots, k\}$, $i \neq j$, is defined as
$$\eta_{i,j}((D, \alpha)) = (D', \alpha),$$
where D' is D after introducing all possible arcs with tail labeled with i and head labeled with j.
- **Relabel** $\rho_{i \to j}(\cdot)$ for $i, j \in \{1, 2, \dots, k\}$, $i \neq j$, is defined as
$$\rho_{i \to j}((D, \alpha)) = (D, \alpha'),$$
where α' is α with all the values i substituted with j.

A *clique expression* is a term that uses operators $\oplus$, $\eta_{i,j}(\cdot)$, $\rho_{i \to j}(\cdot)$, and constants $\iota_1, \iota_2, \dots, \iota_k$ that represent one-vertex graphs with the only vertex labeled with $1, 2, \dots, k$, respectively. In this manner a clique expression *constructs* some labeled digraph (D, α). The cliquewidth of a digraph D (denoted $\mathbf{cw}(D)$) is the minimum number of labels needed in a clique expression that constructs D (with any labeling).

2.3.4 Comparison of the parameters

Lemma 2.13. *For every digraph D, it holds that $\mathbf{pw}(D) \leq 2 \cdot \mathbf{ctw}(D)$. Moreover, given an ordering of $V(D)$ of cutwidth c, one can in $\mathcal{O}(|V(D)|^2)$ time compute a path decomposition of width at most $2c$.*

Proof. We provide a method of construction of a path decomposition of width at most $2c$ from an ordering of $V(D)$ of cutwidth c.

Let $(v_1, v_2, \dots, v_n)$ be the ordering of $V(D)$ of cutwidth c. Let $F \subseteq E(D)$ be the set of edges (v_j, v_i) such that $j > i$; edges from F will be called *back edges*. We now construct a path decomposition $W = (W_1, W_2, \dots, W_n)$ of D by setting

$$W_\ell = \{v_\ell\} \cup \bigcup \{\{v_i, v_j\} \mid i \leq \ell < j \ \wedge \ (v_j, v_i) \in E(D)\}.$$

In other words, for each cut between two consecutive vertices in the order (plus one extra at the end of the ordering) we construct a bag that contains (i) endpoints of all the back edges that are cut by this cut, and (ii) the last vertex before the cut. Observe that $|W_\ell| \leq 2c + 1$ for every $1 \leq \ell \leq n$. It is easy to construct W in $\mathcal{O}(|V(D)|^2)$ time using one scan through the ordering $(v_1, v_2, \ldots, v_n)$. We are left with arguing that W is a path decomposition. Clearly $\bigcup_{i=1}^{n} W_i = V(D)$, so property (i) holds

Consider any vertex v_ℓ and an index $j \neq \ell$, such that $v_\ell \in W_j$. Assume first that $j < \ell$. By the definition of W, there exists an index $i \leq j$ such that $(v_\ell, v_i) \in E(D)$. Existence of this arc implies that v_ℓ has to be contained in every bag between W_j and W_ℓ, by the definition of W. A symmetrical reasoning works also for $j \geq \ell$. We infer that for every vertex v_ℓ the set of bags it is contained in form an interval in the path decomposition. This proves property (ii).

To finish the proof, consider any edge $(v_i, v_j) \in E(G)$. If $i > j$ then $\{v_i, v_j\} \subseteq W_j$, whereas if $i < j$ then $v_i \in W_i$ and $v_j \in W_j$. Thus, property (iii) holds as well. □

Lemma 2.14. *For every semi-complete digraph T, it holds that $\mathbf{cw}(T) \leq \mathbf{pw}(T)+2$. Moreover, given a path decomposition of width p, one can in $\mathcal{O}(|V(T)|^2)$ time compute a clique expression constructing T using $p+2$ labels.*

Proof. We provide a method of construction of a clique expression using $p + 2$ labels from a path decomposition of width p. Let $(W_1, W_2, \ldots, W_r)$ be a path decomposition of T of width p. Without loss of generality we can assume that the given path decomposition is nice. As in a nice path decomposition every vertex is introduced and forgotten exactly once, we have that $r = 2|V(T)| + 1$.

We now build a clique expression using $p+2$ labels that constructs the semi-complete digraph T along the path decomposition. Intuitively, at each step of the construction, every vertex of the bag W_i is assigned a different label between 1 and $p + 1$, while all forgotten vertices are assigned label $p + 2$. If we proceed in this manner, we will end up with the whole digraph T labeled with $p+2$, constructed for the last bag. As we begin with an empty graph, we just need to show what to do in the introduce and forget vertex steps.

Introduce vertex step.

Assume that $W_i = W_{i-1} \cup \{v\}$, i.e., bag W_i introduces vertex v. Note that this means that $|W_{i-1}| \leq p$. As labels from 1 up to $p + 1$ are assigned to vertices of W_{i-1} and there are at most p of them, let q be a label that is not assigned. We perform following operations; their correctness is straightforward.

- perform $\oplus\iota_q$: we introduce the new vertex with label q;
- for each $w \in W_{i-1}$ with label q', perform join $\eta_{q,q'}$ if $(v, w) \in E(D)$ and join $\eta_{q',q}$ if $(w, v) \in E(D)$;
- perform join $\eta_{q,p+2}$ since the new vertex has an outgoing arc to every forgotten vertex.

Forget vertex step.

Assume that $W_i = W_{i-1} \setminus \{w\}$, i.e., bag W_i forgets vertex w. Let $q \in \{1, 2, \ldots, p+1\}$ be the label of w. We just perform relabel operation $\rho_{q \to p+2}$, thus moving w to forgotten vertices. □

2.4 MSO and semi-complete digraphs

In this work we use known results on model checking Monadic Second-Order Logic on (di)graphs of small cliquewidth. In the sequel, $\mathbf{MSO}_1$ is Monadic Second-Order Logic with quantification over subsets of vertices but not of arcs, whereas in $\mathbf{MSO}_2$ we allow also quantification over arc subsets. We will be mostly interested in the tractability of model checking $\mathbf{MSO}_1$ on digraphs of bounded cliquewidth. For completeness, we now briefly recall the definitions of $\mathbf{MSO}_1$ and $\mathbf{MSO}_2$ on digraphs. For more information on the links between these models of logic and fixed-parameter tractability, we refer to the monograph of Courcelle and Engelfriet [14], or to a shorter introduction in the book of Flum and Grohe [25].

The syntax of $\mathbf{MSO}_1$ on digraphs consists of:

- Logical connectives $\vee$, $\wedge$, $\neg$, $\Leftrightarrow$, $\Rightarrow$, with standard semantics.
- Variables for vertices and for subsets of vertices.
- Quantifiers $\forall$, $\exists$ that can be applied to these variables. The semantics is that $\forall_{x \in V}\ \psi$ is true if and only if ψ is true for *every* evaluation of x to a vertex of the digraph, whereas $\exists_{x \in V}\ \psi$ is true if and only if ψ is true for *some* evaluation of x to a vertex of the digraph. The semantics of $\forall_{X \subseteq V}$ and $\exists_{X \subseteq V}$ is defined analogically.
- The following binary relations:
  - $\mathbf{A}(u, v)$, where u, v are vertex variables, and the semantics is that $\mathbf{A}(u, v)$ is true if and only if the arc (u, v) is present in the digraph;
  - $x \in X$, where x is a vertex variable and X is a vertex set variable, with standard semantics;
  - equality of variables.

$\mathbf{MSO}_2$ extends $\mathbf{MSO}_1$ by introducing also variables for arcs and subsets of arcs, and allowing quantification over these variables as well. There is also one additional binary relation $\mathbf{inc}(v, e)$ that checks whether an edge e is incident to a vertex v.

In the undirected setting, it is widely known that model checking $\mathbf{MSO}_2$ is fixed-parameter tractable, when the parameters are the length of the formula and the treewidth of the graph; see e.g. [14, 19, 25]. As far as $\mathbf{MSO}_1$ is concerned, model checking $\mathbf{MSO}_1$ is fixed-parameter tractable, when the parameters are the length of the formula and the cliquewidth of the graph. These results in fact hold not only for undirected graphs, but for structures with binary relations in general (where we consider the Gaifman graph of the structure), so in particular for digraphs. The following result follows from the work of Courcelle, Makowsky and Rotics [15]; we remark that the original paper treats of undirected graphs, but in fact the results hold also in the directed setting (cf. [32, 34, 39]).

Theorem 2.15 ([15])**.** *There exists an algorithm with running time $f(||\varphi||, k) \cdot n^2$ that given an $\mathbf{MSO}_1$ formula φ checks whether φ is satisfied in a digraph G on n vertices, given together with a clique expression using at most k labels constructing it.*

Lemma 2.14 asserts that the cliquewidth of a semi-complete digraph is bounded by its pathwidth plus 2. Moreover, the proof gives explicit construction of the corresponding expression. Hence, the following meta-theorem follows as an immediate corollary.

Theorem 2.16. *There exists an algorithm with running time $f(||\varphi||, p) \cdot n^2$ that given an $\mathbf{MSO}_1$ formula φ checks whether φ is satisfied in a semi-complete digraph T on n vertices, given together with a path decomposition of width p.*

We note that by pipelining Lemmas 2.13 and 2.14 one can show that an analogous result holds also for cutwidth.

It is tempting to conjecture that the tractability result for $\mathbf{MSO}_1$ and pathwidth or cutwidth could be extended also to $\mathbf{MSO}_2$, as the decompositions resemble path decompositions in the undirected setting. However, this is unfortunately not true. In particular, as it was shown in [44], there exists a constant-size $\mathbf{MSO}_2$ formula ψ, such that checking whether ψ is true in a semi-complete digraph of constant cutwidth and pathwidth is NP-hard.

3 The obstacle zoo

In this section we describe the set of obstacles used by the algorithms. We begin with *jungles*, the original obstacle introduced by Fradkin and Seymour [31], and their enhanced versions that will be used extensively in this paper, namely *short jungles.* It appears that the enhancement enables us to construct large topological subgraph, minor or immersion models in short jungles in a greedy manner, and this observation is the key to trimming the running times for containment tests. We then recall the notion of a *triple* that is an important concept introduced by Fradkin and Seymour [31], and which we will also use for irrelevant vertex rules. Finally, we continue with further obstacles that will be used in the algorithms: degree and matching tangles for pathwidth, and backward tangles for cutwidth. Each time we describe one of these obstacles, we prove two lemmas. The first asserts that existence of the structure is indeed an obstacle for having small width, while the second shows that one can constructively find an appropriate short jungle in a sufficiently large obstacle.

3.1 Jungles and short jungles

Definition 3.1. *Let T be a semi-complete digraph and k be an integer. A k-*jungle *is a set $X \subseteq V(T)$ such that* (i) $|X| = k$*;* (ii) *for every $v, w \in X$, $v \neq w$, either $(v, w) \in E(T)$ or there are k internally vertex-disjoint paths from v to w.*

It is easy to observe that existence of a $(k + 1)$-jungle is an obstacle for admitting a path decomposition of width smaller than k, as in such a decomposition there would necessarily be a bag containing all the vertices of the jungle. The work of Fradkin and Seymour [31] essentially says that containing a large jungle is the only reason for not admitting a decomposition of small width: if $\mathbf{pw}(T) \geq f(k)$ for some function f (that is actually quadratic), then T must necessarily contain a k-jungle. In this paper we strenghten this result by providing linear bounds on function f, and moreover showing that in the obtained jungle the paths between vertices may be assumed to be short, as in the definition below.

Definition 3.2. *Let T be a semi-complete digraph and k, d be integers. A (k, d)-*short (immersion) jungle *is a set $X \subseteq V(T)$ such that* (i) $|X| \geq k$*;* (ii) *for every $v, w \in X$, $v \neq w$, there are k internally vertex-disjoint (edge-disjoint) paths from v to w of length at most d.*

We remark that in this definition we treat a path as a subdigraph. Thus, among the k vertex-disjoint paths from v to w only at most one can be of length 1. We remark also that in all our algorithms, every short jungle is constructed and stored together with corresponding families of k paths for each pair of vertices.

The restriction on the length of the paths enables us to construct topological subgraph and immersion models in short jungles greedily. This is the most useful advantage of short jungles over jungles, as it enables us to reduce the time complexity of containment tests to single-exponential.

Lemma 3.3. *If a digraph T contains a (dk, d)-short (immersion) jungle for some $d > 1$, then it admits every digraph S with $|S| \leq k$ as a topological subgraph (as an immersion).*

Proof. Firstly, we prove the lemma for short jungles and topological subgraphs. Let X be the short jungle whose existence is assumed. We construct the expansion greedily. As images of vertices of S we put arbitrary $|V(S)|$ vertices of X. Then we construct paths being images of arcs in S; during each construction we use at most $d - 1$ new vertices of the digraph for the image. While constructing the i-th path, which has to lead from v to w, we consider dk vertex-disjoint paths of length d from v to w. So far we used at most $k + (i - 1)(d - 1) < dk$

vertices for images, so at least one of these paths does not traverse any used vertex. Hence, we can safely use this path as the image and proceed; note that thus we use at most $d-1$ new vertices.

Secondly, we prove the lemma for short immersion jungles and immersions. Let X be the short immersion jungle whose existence is assumed. We construct the immersion greedily. As images of vertices of S we put arbitrary $|V(S)|$ vertices of X. Then we construct paths being images of arcs in S; during each construction we use at most d new arcs of the digraph for the image. While constructing the i-th path, which has to lead from v to w, we consider dk edge-disjoint paths of length d from v to w. So far we used at most $(i-1)d < dk$ arcs, so at least one of these paths does not contain any used arc. Hence, we can safely use this path as the image and proceed; note that thus we use at most d new arcs. □

We now prove the analogue of Lemma 3.3 for minors. We remark that unlike Lemma 3.3, the proof of the following lemma is nontrivial.

Lemma 3.4. *If a digraph T contains a $(2dk+1,d)$-short jungle for some $d>1$, then it admits every digraph S with $|S|\leq k$ as a minor.*

Proof. Let S' be a digraph constructed as follows: we take S and whenever for some vertices v,w arc (v,w) exists but (w,v) does not, we add also the arc (w,v). We have that $|S'|\leq 2|S|\leq 2k$, so since T contains a $(2dk+1,d)$-jungle, by Lemma 3.3 we infer that T contains S' as a topological subgraph. Moreover, since for every two vertices of this jungle there is at most one path of length 1 between them, in the construction of Lemma 3.3 we may assume that we always use one of the paths of length longer than 1. Hence, we can assume that all the paths in the constructed expansion of S' in T are of length longer than 1.

Let η' be the constructed expansion of S' in T. Basing on η', we build a minor model η of S' in T; since S is a subdigraph of S', the lemma will follow. We start with $\eta(v)=\{\eta'(v)\}$ for every $v\in V(S')$ and gradually add vertices to each $\eta(v)$.

Consider two vertices $v,w\in V(S')$ such that $(v,w),(w,v)\in E(S')$. We have then two vertex disjoint paths $P=\eta'((v,w))$ and $Q=\eta'((w,v))$ that lead from $\eta'(v)$ to $\eta'(w)$ and vice versa. We know that P and Q are of length at least 2. Moreover, without loss of generality we may assume that for any pair of vertices (x,y) on P that are (i) not consecutive, (ii) y appears on P_1 after x, and (iii) $(x,y)\neq(\eta'(v),\eta'(w))$, we have that $(y,x)\in E(T)$. Indeed, otherwise we would have that $(x,y)\in E(T)$ and path P could be shortcutted using arc (x,y) without spoiling the property that it is longer than one. We can assume the same for the path Q.

Assume first that one of the paths P,Q is in fact of length greater than 2. Assume without loss of generality that it is P, the construction for Q is symmetric. Let p_1 and p_2 be the first and the last internal vertex of P, respectively. By our assumptions about nonexistence of shortcuts on P and about $|P|>2$, we know that $p_1\neq p_2$, $(p_2,\eta'(v))\in E(T)$ and $(\eta'(w),p_1)\in E(T)$. Observe that the subpath of P from $\eta'(v)$ to p_2, closed by the arc $(p_2,\eta'(v))$ forms a directed cycle. Include the vertex set of this cycle into $\eta(v)$. Observe that thus we obtain an arc $(p_2,\eta'(w))$ from $\eta(v)$ to $\eta(w)$, and an arc $(\eta'(w),p_1)$ from $\eta(w)$ to $\eta(v)$.

Now assume that both of the paths P,Q are of length exactly 2, that is, $P=\eta'(v)\to p\to\eta'(w)$ for some vertex p, and $Q=\eta'(w)\to q\to\eta'(v)$ for some vertex $q\neq p$. Since T is semi-complete, at least one of arcs $(p,q),(q,p)$ exists. Assume without loss of generality that $(p,q)\in E(T)$; the construction in the second case is symmetric. Note that $\eta'(v)\to p\to q\to\eta'(v)$ is a directed cycle; include the vertex set of this cycle into $\eta(v)$. Observe that thus we obtain an arc $(p,\eta'(w))$ from $\eta(v)$ to $\eta(w)$, and an arc $(\eta'(w),q)$ from $\eta(w)$ to $\eta(v)$.

Concluding, for every $v\in V(T)$ the final $\eta(v)$ consists of $\eta'(v)$ plus vertex sets of directed cycles that pairwise meet only in $\eta'(v)$. Thus, $T[\eta(v)]$ is strongly connected for each $v\in V(T)$.

Moreover, for every pair of vertices $v, w \in V(S')$ such that $(v, w), (w, v) \in E(S')$, we have pointed out an arc from $\eta(v)$ to $\eta(w)$ and from $\eta(w)$ to $\eta(v)$. Hence, η is a minor model of S'. $\square$

3.2 Triples

We now recall the notion of a *triple*, an obstacle extensively used in the approach of Fradkin and Seymour [31].

Definition 3.5. *Let T be a semi-complete digraph. A triple of pairwise disjoint subsets (A, B, C) is called a k-triple if $|A| = |B| = |C| = k$ and there exist orderings $(a_1, \ldots, a_k)$, $(b_1, \ldots, b_k)$, $(c_1, \ldots, c_k)$ of A, B, C, respectively, such that for all indices $1 \leq i, j \leq k$ we have $(a_i, b_j), (b_i, c_j) \in E(T)$ and for each index $1 \leq i \leq k$ we have $(c_i, a_i) \in E(T)$.*

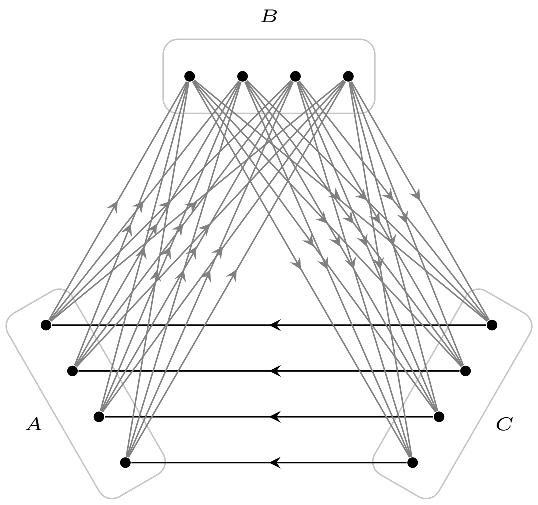

Figure 3: A 4-triple.

The main observation of [31] is that for some function f, if a semi-complete digraph contains an $f(k)$-jungle, then it contains also a k-triple [31, (2.6)]. Moreover, if it contains a k-triple, then every digraph of size at most k is topologically contained in this triple [31, (1) in the proof of (1.1)]. We remark that Fradkin and Seymour actually attribute this observation to Chudnovsky, Scott, and Seymour. The following lemma is an algorithmic version of the aforementioned observation and will be needed in our algorithms. The proof closely follows the lines of argumentation contained in [31]; we include it for the sake of completeness.

Lemma 3.6. *There exists an elementary function f, such that for every $k \geq 1$ and every semi-complete graph T on n vertices, given together with a $f(k)$-jungle in it, it is possible to construct a k-triple in T in time $\mathcal{O}(n^3 \log n)$.*

Proof. For an integer k, let $R(k, k)$ denote the Ramsey number, that is the smallest integer such that every red-blue coloring of the edges of the complete graph on $R(k, k)$ vertices contains a monochromatic clique of size k. By the theorem of Erdős and Szekeres [21], $R(k, k) \leq (1 + o(1))\frac{4^{k-1}}{\sqrt{\pi k}}$. For $k \geq 1$, we define function the function f as

$$f(k) = 2^{12 \cdot 2^{R(2k,2k)}}.$$

Moreover, let $r = R(2k, 2k)$, $s = 2^r$, and $m = 2^{12s} = f(k)$.

We say that a semi-complete digraph is *transitive* if it contains a transitive tournament as a subdigraph. Every tournament on m vertices contains a transitive tournament on $\log_2 m$ vertices as a subdigraph. Also an induced acyclic subdigraph of an oriented (where there is at most one directed arc between a pair of vertices) m-vertex digraph with $\log_2 m$ vertices can be found in time $\mathcal{O}(m^2 \log m)$ [46]. This algorithm can be modified into an algorithm finding a transitive semi-complete digraph in semi-complete digraphs by removing first all pairs of oppositely directed arcs, running the algorithm for oriented graphs, and then adding some of the deleted arcs to turn the acyclic digraph into a transitive semi-complete digraph. Thus, if X_0 is an m-jungle in T, then X_0 contains a subset X that is a $12s$-jungle in T and that induces a transitive semi-complete digraph. Moreover, such a set X can be found in time $\mathcal{O}(n^2 \log n)$.

The next step in the proof of Fradkin and Seymour is to partition the set X into parts X_1 and X_2 of size $6s$ each such that X_1 is complete to X_2, i.e., for each $x_1 \in X_1$ and $x_2 \in X_2$ we have $(x_1, x_2) \in E(T)$. Such a partition of the vertex set of the transitive semi-complete digraph can be easily found in time $\mathcal{O}(|X|^2)$. Because X is a $12s$-jungle in T, there are at least $6s$ vertex-disjoint paths from X_2 to X_1 in T. Indeed, otherwise by Menger's theorem there would be a separation (A, B) of order less than $6s$ such that $X_2 \subseteq A$ and $X_1 \subseteq B$, and such a separation would separate some vertex from X_1 from some vertex of X_2, contradicting existence of $6s$ internally vertex-disjoint paths between these two vertices. Let R be a minimal induced subdigraph of T such that $X \subseteq V(R)$ and there are $6s$ vertex-disjoint paths from X_2 to X_1 in R. Such a minimal subgraph R can be found in time $\mathcal{O}(n^3 \log n)$ by repeatedly removing vertices $v \in V(T) \setminus X$ if there are $6s$ vertex-disjoint paths from X_2 to X_1 in $V(T) \setminus \{v\}$. As the subroutine for finding the paths we use the classical Ford-Fulkerson algorithm, where we finish the computation after finding $6s$ paths. Hence, the running time one application of this algorithm is $\mathcal{O}(sn^2)$. As we make at most n tests and $s = \mathcal{O}(\log n)$, the claimed bound on the runtime follows.

Let $P_1, P_2, \ldots, P_{6s}$ be vertex-disjoint paths from X_2 to X_1 in R. The arguments given by Fradkin and Seymour prove that the set of vertices Q formed by the first two and the last two vertices of these $6s$ paths contains a k-triple. Thus, by checking every triple of subsets of Q of size k in time polynomial in k, we can find a k-triple. This step takes time $\mathcal{O}\left(\binom{24s}{k}^3 k^{\mathcal{O}(1)}\right) = n^{o(1)}$, as $s = \mathcal{O}(\log n)$ and $k = \mathcal{O}(\log \log n)$. □

3.3 Degree tangles

The degree tangle is intuitively a concentration of vertices with very similar outdegrees. Surprisingly, a large degree tangle forms already an obstacle for admitting a path decomposition of small width. This observation is the main idea behind the degree ordering approach that we use in this paper.

Definition 3.7. *Let T be a semi-complete digraph and k, ℓ be integers. A (k, ℓ)-*degree tangle *is a set $X \subseteq V(T)$ such that* (i) *$|X| \geq k$;* (ii) *for every $v, w \in X$ we have $|d^+(v) - d^+(w)| \leq \ell$.*

Lemma 3.8. *Let T be a semi-complete digraph. If T contains a $(4k+2, k)$-degree tangle X, then $\mathbf{pw}(T) > k$.*

Proof. For the sake of contradiction, assume that T admits a (nice) path decomposition W of width at most k. Let $\alpha = \min_{v \in X} d^+(v)$ and $\beta = \max_{v \in X} d^+(v)$; we know that $\beta - \alpha \leq k$. Let $(A, B) = W[\alpha]$. Recall that $(A, B) = W[\alpha]$ is any separation in the separation chain corresponding to W in the sense of Lemma 2.12 such that $|A| = \alpha$. We know that $|A \cap B| \leq k$ and $|A| = \alpha$.

Firstly, observe that $X \cap (A \setminus B) = \emptyset$. This follows from the fact that vertices in $A \setminus B$ can have outneighbors only in A, so their outdegrees are upper bounded by $|A| - 1 = \alpha - 1$.

Secondly, $|X \cap (A \cap B)| \leq k$ since $|A \cap B| \leq k$.

Thirdly, we claim that $|X \cap (B \setminus A)| \leq 3k + 1$. Assume otherwise. Consider subdigraph $T[X \cap (B \setminus A)]$. Since this is a subdigraph of a semi-complete digraph of pathwidth at most k, it has also pathwidth at most k. By Lemma 2.10 we infer that there exists a vertex $v \in X \cap (B \setminus A)$ whose indegree in $T[X \cap (B \setminus A)]$ is at most k. Since $T[X \cap (B \setminus A)]$ is semi-complete and $|X \cap (B \setminus A)| \geq 3k+2$, we infer that the outdegree of v in $T[X \cap (B \setminus A)]$ is at least $2k+1$. As (A, B) is a separation and T is semi-complete, all the vertices of $A \setminus B$ are also outneighbors of v in T. Note that $|A \setminus B| = |A| - |A \cap B| \geq \alpha - k$. We infer that v has at least $\alpha - k + 2k + 1 = \alpha + k + 1 > \beta$ outneighbors in T, which is a contradiction with $v \in X$.

Summing up the bounds we get $4k + 2 \leq |X| \leq k + 3k + 1 = 4k + 1$, a contradiction. □

Lemma 3.9. *Let T be a semi-complete digraph and let X be a $(26k, k)$-degree tangle in T. Then X contains a $(k, 3)$-short jungle which can be found in $\mathcal{O}(k^3 n^2)$ time, where $n = |V(T)|$.*

Proof. We present a proof of the existential statement. The proof can be easily turned into an algorithm finding the jungle; during the description we make remarks at the places where it may be non-trivial to observe how the algorithm should perform to achieve the promised running-time guarantee.

By possibly trimming X, assume without loss of generality that $|X| = 26k$. Take any $v, w \in X$. We either find a $(k, 3)$-short jungle in X explicitly, or find k vertex-disjoint paths from v to w of length at most 3. If for no pair v, w an explicit short jungle is found, we conclude that X is a $(k, 3)$-short jungle itself.

Let us consider four subsets of $V(T) \setminus \{v, w\}$:

- $V^{++} = (N^+(v) \cap N^+(w)) \setminus \{v, w\}$,
- $V^{+-} = (N^+(v) \cap N^-(w)) \setminus \{v, w\}$,
- $V^{-+} = (N^-(v) \cap N^+(w)) \setminus \{v, w\}$,
- $V^{--} = (N^-(v) \cap N^-(w)) \setminus \{v, w\}$.

Clearly, $|V^{++}| + |V^{-+}| + |V^{+-}| + |V^{--}| \geq n - 2$. Note that equality holds for the tournament case — in this situation these four subsets form a partition of $V(T) \setminus \{v, w\}$.

If $|V^{+-}| \geq k$, then we already have k vertex-disjoint paths of length 2 from v to w. Assume then that $|V^{+-}| < k$.

Observe that $d^+(v) \leq |V^{++}| + |V^{+-}| + 1$ and $d^+(w) \geq |V^{++}| + |V^{-+} \setminus V^{++}|$. Since $v, w \in X$, we have that $d^+(w) - d^+(v) \leq k$, so

$$|V^{-+} \setminus V^{++}| \leq d^+(w) - |V^{++}| \leq k + d^+(v) - |V^{++}| \leq k + 1 + |V^{+-}| \leq 2k.$$

Let $A = V^{++} \setminus V^{+-}$ and $B = V^{--} \setminus V^{+-}$. Note that A and B are disjoint, since $V^{++} \cap V^{--} \subseteq V^{+-}$. Let H be a bipartite graph with bipartition (A, B), such that for $a \in A$ and $b \in B$ we have $ab \in E(H)$ if and only if $(a, b) \in E(T)$. Every edge ab of H gives raise to a path $v \to a \to b \to w$ of length 3 from v to w. Hence, if we could find a matching of size k in H, then this matching would form a family of k vertex disjoint paths of length at most 3 from v to w. Note that testing existence of such a matching can be done in $\mathcal{O}(kn^2)$ time, as we can run the algorithm finding an augmenting path at most k times.

Assume then that such a matching does not exist. By Kőnig's theorem we can find a vertex cover C of H of cardinality smaller than k; again, this can be found in $\mathcal{O}(kn^2)$ time. As

$A \cup B \cup (V^{-+} \setminus V^{++}) \cup V^{+-} = V(T) \setminus \{v, w\}$ while $(V^{-+} \setminus V^{++}) \cup V^{+-}$ contains at most than $3k-1$ vertices in total, $A \cup B$ must contain at least $(26k-2)-(3k-1) = 23k-1$ vertices from X. We consider two cases: either $|A \cap X| \geq 16k$, or $|B \cap X| \geq 7k$.

Case 1. In the first case, consider set $Y_0 = X \cap (A \setminus C)$. Since $|A \cap X| \geq 16k$ and $|A \cap C| < k$, we have that $|Y_0| > 15k$. Let Y be any subset of Y_0 of size $15k$. Take any vertex $y \in Y$ and consider, where its outneighbors can lie. These outneighbors can be either in $\{v\}$ (at most 1 of them), in $(V^{-+} \setminus V^{++}) \cup V^{+-}$ (less than $3k$ of them), in $B \cap C$ (at most k of them), or in A. As $d^+(v) \geq |A|$ and $v, y \in X$, we have that $d^+(y) \geq |A| - k$. We infer that y must have at least $|A| - 5k$ outneighbors in A. As $|Y| = 15k$, we have that y has at least $10k$ outneighbors in Y.

Note that in the tournament case we would be already finished, as this lower bound on the outdegree would imply also an upper bound on indegree, which would contradict the fact that $T[Y]$ contains a vertex of indegree at least $\frac{|Y|-1}{2}$. This also shows that in the tournament setting a stronger claim holds that in fact X is a $(k,3)$-jungle itself. In the semi-complete setting, however, we do not have any contradiction yet. In fact no contradiction is possible as the stronger claim is no longer true. To circumvent this problem, we show how to find an explicit $(k,3)$-short jungle within Y.

Observe that the sum of outdegrees in $T[Y]$ is at least $10k \cdot 15k = 150k^2$. We claim that the number of vertices in Y that have indegrees at least $6k$ is at least k. Otherwise, the sum of indegrees would be bounded by $15k \cdot k + 6k \cdot 14k = 99k^2 < 150k^2$ and the sums of the indegrees and of the outdegrees would not be equal. Let Z be any set of k vertices in Y that have indegrees at least $6k$ in $T[Y]$. Take any $z_1, z_2 \in Z$ and observe that in $T[Y]$ the set of outneighbors of z_1 and the set of inneighbors of z_2 must have intersection of size at least k, as $d^+_{T[Y]}(z_1) \geq 10k$, $d^-_{T[Y]}(z_2) \geq 6k$ and $|Y| = 15k$. Through these k vertices one can rout k vertex-disjoint paths from z_1 to z_2, each of length 2. Hence, Z is the desired $(k,3)$-short jungle.

Case 2. This case will be similar to the previous one, with the exception that we only get a contradiction: there is no subcase with finding an explicit smaller jungle. Consider set $Y_0 = X \cap (B \setminus C)$. Since $|B \cap X| \geq 7k$ and $|B \cap C| < k$, we have that $|Y_0| > 6k$. Let Y be any subset of Y_0 of size $6k+1$. Take any vertex $y \in Y$ that has outdegree at least $3k$ in $T[Y]$ (since $|Y| = 6k+1$, such a vertex exists), and consider its outneighbors. As $y \notin C$ we have that all the vertices of $A \setminus C$ are the outneighbors of y (more than $|A| - k$ of them), and there are at least $3k$ outneighbors within B. Hence $d^+(y) > |A| + 2k$. On the other hand, the outneighbors of v have to lie inside $A \cup V^{+-} \cup \{w\}$, so $d^+(v) \leq |A \cup V^{+-} \cup \{w\}| \leq |A| + k$. We infer that $d^+(y) - d^+(v) > k$, which is a contradiction with $v, y \in X$. □

3.4 Matching tangles

Definition 3.10. *Let T be a semi-complete digraph and k, ℓ be integers. A (k,ℓ)-*matching tangle *is a pair of disjoint subsets $X, Y \subseteq V(T)$ such that* (i) *$|X| = |Y| = k$;* (ii) *there exists a matching from X to Y, i.e., there is a bijection $f : X \to Y$ such that $(v, f(v)) \in E(T)$ for all $v \in X$;* (iii) *for every $v \in X$ and $w \in Y$ we have that $d^+(w) > d^+(v) + \ell$.*

Lemma 3.11. *Let T be a semi-complete digraph. If T contains a $(k+1,k)$-matching tangle (X,Y), then $\mathbf{pw}(T) > k$.*

Proof. For the sake of contradiction assume that T has a (nice) path decomposition W of width at most k. Let $\alpha = \min_{w \in Y} d^+(w)$ and let $(A,B) = W[\alpha]$. Recall that $|A| = \alpha$ and $|A \cap B| \leq k$.

Firstly, we claim that $X \subseteq A$. Assume otherwise that there exists some $v \in (B \setminus A) \cap X$. Note that all the vertices of $A \setminus B$ are outneighbors of v, so $d^+(v) \geq |A| - k = \alpha - k$. Hence $d^+(v) \geq d^+(w) - k$ for some $w \in Y$, which is a contradiction.

Secondly, we claim that $Y \subseteq B$. Assume otherwise that there exists some $w \in (A \setminus B) \cap Y$. Then all the outneighbors of w are in A, so there is less than α of them. This is a contradiction with the definition of α.

As $|A \cap B| \leq k$ and there are $k+1$ disjoint pairs of form $(v, f(v)) \in E(T)$ for $v \in X$, we conclude that there must be some $v \in X$ such that $v \in A \setminus B$ and $f(v) \in B \setminus A$. This contradicts the fact that (A, B) is a separation. □

Lemma 3.12. *Let T be a semi-complete digraph and let (X, Y) be a $(5k, 3k)$-matching tangle in T. Then Y contains a $(k, 4)$-short jungle which can be found in $\mathcal{O}(k^3 n)$ time, where $n = |V(T)|$.*

Proof. We present the proof of the existential statement; all the steps of the proof are easily constructive and can be performed within the claimed complexity bound.

Let Z be the set of vertices of Y that have indegrees at least $k+1$ in $T[Y]$. We claim that $|Z| \geq k$. Otherwise, the sum of indegrees in $T[Y]$ would be at most $k \cdot 5k + 4k \cdot k = 9k^2 < \binom{5k}{2}$, so the total sum of indegrees would be strictly smaller than the number of arcs in the digraph. It remains to prove that Z is a $(k, 4)$-short jungle.

Take any $v, w \in Z$; we are to construct k vertex-disjoint paths from v to w, each of length at most 4. Let $R_0 = N^-_{T[Y]}(w) \setminus \{v\}$. Note that $|R_0| \geq k$, hence let R be any subset of R_0 of cardinality k and let $P = f^{-1}(R)$. We are to construct k vertex-disjoint paths of length 2 connecting v with every vertex of P and not passing through $P \cup R \cup \{w\}$. By concatenating these paths with arcs of the matching f between P and R and arcs leading from R to w, we obtain the family of paths we look for.

The paths from v to P are constructed in a greedy manner, one by one. Each path construction uses exactly one vertex outside $P \cup R$. Let us take the next, i-th vertex $p \in P$. As $d^+(v) > d^+(p) + 3k$, by Lemma 2.2 there exist at least $3k$ vertices in T that are both outneighbors of v and inneighbors of p. At most $2k$ of them can be inside $P \cup R$, at most $i - 1 \leq k - 1$ of them were used for previous paths, so there is at least one that is still unused; let us denote it by q. If in fact $q = w$, we build a path of length 1 directly from v to w thus ignoring vertex p; otherwise we can build the path of length 2 from v to p via q and proceed to the next vertex of P. □

3.5 Backward tangles

Definition 3.13. *Let T be a semi-complete digraph and k be an integer. A k-*backward tangle *is a partition (X, Y) of $V(T)$ such that* (i) *there exist at least k arcs directed from X to Y;* (ii) *for every $v \in X$ and $w \in Y$ we have that $d^+(w) \geq d^+(v)$.*

Lemma 3.14. *Let T be a semi-complete digraph. If T contains an $(m+1)$-backward tangle (X, Y) for $m = 64k^2 + 18k + 1$, then $\mathbf{ctw}(T) > k$.*

Proof. For the sake of contradiction, assume that $V(T)$ admits an ordering π of width at most k. Let α be the largest index such that $(X_\alpha, Y_\alpha) = (\pi[\alpha], V(T) \setminus \pi[\alpha])$ satisfies $Y \subseteq Y_\alpha$. Similarly, let β be the smallest index such that $(X_\beta, Y_\beta) = (\pi[\beta], V(T) \setminus \pi[\beta])$ satisfies $X \subseteq X_\beta$. Note that $|E(X_\alpha, Y_\alpha)|, |E(X_\beta, Y_\beta)| \leq k$. Observe also that $\alpha \leq \beta$; moreover, $\alpha < |V(T)|$ and $\beta > 0$, since X, Y are non-empty.

Let $(X_{\alpha+1}, Y_{\alpha+1}) = (\pi[\alpha+1], V(T) \setminus \pi[\alpha+1])$. By the definition of α there is a unique vertex $w \in X_{\alpha+1} \cap Y$. Take any vertex $v \in V(T)$ and suppose that $d^+(w) > d^+(v) + (k+1)$. By Lemma 2.2, there exist $k+1$ vertex-disjoint paths of length 2 from w to v. If v was in $Y_{\alpha+1}$, then each of these paths would contribute at least one arc to the set $E(X_{\alpha+1}, Y_{\alpha+1})$, contradicting the fact that $|E(X_{\alpha+1}, Y_{\alpha+1})| \leq k$. Hence, every such v belongs to $X_{\alpha+1}$ as well. By Lemma 3.8 we have that the number of vertices with outdegrees in the interval $[d^+(w) - (k+1), d^+(w)]$ is bounded by $8k+1$, as otherwise they would create a $(8k+2, 2k)$-degree tangle, implying by Lemma 3.8 that $\mathbf{pw}(T) > 2k$ and, consequently by Lemma 2.13, that $\mathbf{ctw}(T) > k$ (here note that for $k = 0$ the lemma is trivial). As $X_\alpha = X_{\alpha+1} \setminus \{w\}$ is disjoint with Y and all the vertices of X have degrees at most $d^+(w)$, we infer that $|X \setminus X_\alpha| \leq 8k + 1$.

A symmetrical reasoning shows that $|Y \setminus Y_\beta| \leq 8k + 1$. Now observe that

$$\begin{aligned} |E(X,Y)| &\leq |E(X_\alpha, Y)| + |E(X, Y_\beta)| + |E(X \setminus X_\alpha, Y \setminus Y_\beta)| \\ &\leq |E(X_\alpha, Y_\alpha)| + |E(X_\beta, Y_\beta)| + |E(X \setminus X_\alpha, Y \setminus Y_\beta)| \\ &\leq k + k + (8k+1)^2 = 64k^2 + 18k + 1. \end{aligned}$$

This is a contradiction with (X, Y) being an $(m+1)$-backward tangle. □

Lemma 3.15. *Let T be a semi-complete digraph and let (X, Y) be an m-backward tangle in T for $m = 109^2k^2$. Then X or Y contains a $(k, 4)$-short immersion jungle which can be found in $\mathcal{O}(k^3n^2)$ time, where $n = |V(T)|$.*

Proof. We present the proof of the existential statement; all the steps of the proof are easily constructive and can be performed within the claimed complexity bound.

Let $P_0 \subseteq X$ and $Q_0 \subseteq Y$ be the sets of heads and of tails of arcs from $E(X, Y)$, respectively. As $|E(X,Y)| \leq |P_0| \cdot |Q_0|$, we infer that $|P_0| \geq 109k$ or $|Q_0| \geq 109k$. Here we consider the first case; the reasoning in the second one is symmetrical.

Let P_1 be the set of vertices in P_0 that have outdegree at least $\alpha - 4k$, where $\alpha = \min_{w \in Y} d^+(w)$; note that $\alpha \geq \max_{v \in X} d^+(v)$ by the definition of a backward tangle. If there were more than $104k$ of them, they would create a $(104k, 4k)$-degree tangle, which due to Lemma 3.9 contains a $(4k, 3)$-short jungle, which is also a $(k, 4)$-short immersion jungle. Hence, we can assume that $|P_1| < 104k$. Let P be any subset of $P_0 \setminus P_1$ of size $5k$. We know that for any $v \in P$ and $w \in Y$ we have that $d^+(w) > d^+(v) + 4k$.

Consider semi-complete digraph $T[P]$. We have that the number of vertices with outdegrees at least k in $T[P]$ is at least k, as otherwise the sum of outdegrees in $T[P]$ would be at most $k \cdot 5k + 4k \cdot k = 9k^2 < \binom{5k}{2}$, so the sum of outdegrees would be strictly smaller than the number of arcs in the digraph. Let Z be an arbitrary set of k vertices with outdegrees at least k in $T[P]$. We prove that Z is a $(k, 4)$-short immersion jungle.

Let us take any $v, w \in Z$; we are to construct k edge-disjoint paths from v to w of length 4. Since the outdegree of v in $T[P]$ is at least k, as the first vertices on the paths we can take any k outneighbors of v in P; denote them $v_1^1, v_2^1, \ldots, v_k^1$. By the definition of P_0, each v_i^1 is incident to some arc from $E(X,Y)$. As the second vertices on the paths we choose the heads of these arcs, denote them by v_i^2, thus constructing paths $v \to v_i^1 \to v_i^2$ of length 2 for $i = 1, 2, \ldots, k$. Note that all the arcs used for constructions so far are pairwise different.

We now consecutively finish paths $v \to v_i^1 \to v_i^2$ using two more arcs in a greedy manner. Consider path $v \to v_i^1 \to v_i^2$. As $v_i^2 \in Y$ and $w \in P$, we have that $d^+(v_i^2) > d^+(w) + 4k$. Hence, by Lemma 2.2 we can identify $4k$ paths of length 2 leading from v_i^2 to w. At most $2k$ of them contain an arc that was used in the first phase of the construction (two first arcs of the paths), and at most $2(i-1) \leq 2k-2$ of them can contain an arc used when finishing previous paths. This leaves us at least one path of length 2 from v_i^2 to w with no arc used so far, which we can use to finish the path $v \to v_i^1 \to v_i^2$. □

4 Algorithms for computing pathwidth

In this section we present approximation and exact algorithms for pathwidth and approximation algorithm for cutwidth. Both of the algorithms employ the technique of sliding through the outdegree ordering of the vertices using a window of small width, and maintaining some coverage of arcs that jump over the window. This coverage can be expressed as a vertex cover of an auxiliary bipartite graph. Therefore, we start our considerations by presenting two ways of selecting a vertex cover in a bipartite graph, called *subset selectors*. The first subset selector, based on the matching theory, will be used in the approximation algorithm, while the second, based on the classical kernel for the VERTEX COVER problem of Buss [8], will be used in the exact algorithm. Armed with our understanding of the introduced subset selectors, we then proceed to the description of the algorithms.

4.1 Subset selectors for bipartite graphs

In this subsection we propose a formalism for expressing selection of a subset of vertices of a bipartite graph. Let $\mathcal{B}$ be the class of undirected bipartite graphs with fixed bipartition, expressed as triples: left side, right side, the edge set. Let $\mu(G)$ be the size of a maximum matching in G.

Definition 4.1. *A function f defined on $\mathcal{B}$ is called a* subset selector *if $f(G) \subseteq V(G)$ for every $G \in \mathcal{B}$. A* reversed *subset selector f^{rev} is defined as $f^{\mathrm{rev}}((X,Y,E)) = f((Y,X,E))$. We say that subset selector f is*

- *a* vertex cover selector *if $f(G)$ is a vertex cover of G for every $G \in \mathcal{B}$, i.e., every edge of G has at least one endpoint in $f(G)$;*
- symmetric *if $f = f^{\mathrm{rev}}$;*
- monotonic *if for every graph $G = (X,Y,E)$ and its subgraph $G' = G \setminus w$ where $w \in Y$, we have that $f(G) \cap X \supseteq f(G') \cap X$ and $f(G) \cap (Y \setminus \{w\}) \subseteq f(G') \cap Y$.*

For our purposes, the goal is to find a vertex cover selector that is at the same time reasonably small in terms of $\mu(G)$, but also monotonic. As will become clear in Section 4.2, only monotonic vertex cover selectors will be useful for us, because monotonicity is essential for obtaining a correct path decomposition where for every vertex the set of bags containing it forms an interval. The following observation expresses, how monotonic subset selectors behave with respect to modifications of the graph; a reader well-familiar with constructing various graph decompositions will probably already see from its statement why monotonicity is so important for us. By addition of a vertex we mean adding a new vertex to the vertex set, together with an arbitrary set of edges connecting it to the old ones.

Lemma 4.2. *Assume that f and f^{rev} are monotonic subset selector and let $G = (X,Y,E)$ be a bipartite graph.*

- *If $v \in f(G) \cap Y$ then v stays chosen by f after any sequence of additions of vertices to the left side and deletions of vertices (different from v) from the right side.*
- *If $v \in X \setminus f(G)$ then v stays not chosen by f after any sequence of additions of vertices to the left side and deletions of vertices from the right side.*

Proof. For both claims, staying (not) chosen after a deletion on the right side follows directly from the definition of monotonicity of f. Staying (not) chosen after an addition on the left side follows from considering deletion of the newly introduced vertex and monotonicity of f^{rev}. $\square$

4.1.1 The matching selector

In this section we define the subset selector that will be used for the approximation algorithm for pathwidth. Throughout this section we assume reader's knowledge of basic concepts and definitions from the classical matching theory (see [18] or [45] for reference). However, we remind the most important facts in the beginning in order to establish notation.

For a bipartite graph $G = (X, Y, E)$ with a matching M, we say that a walk W is an *alternating walk for M*, if

- W starts in a vertex that is unmatched in M;
- edges from M and outside M appear on W alternately, beginning with an edge outside M.

Whenever M is clear from the context, we omit it. If W is in fact a simple path, we say that W is an *alternating path*. A simple shortcutting arguments show that every alternating walk from v to w has an alternating path from v to w as a subsequence. Thus for every vertex $v \notin V(M)$, the set of vertices reachable by alternating walks from v is the same as the set of vertices reachable by alternating paths from v.

Assume we are given a matching M and an alternating path W with respect to M, such that either W is of even length, or the last vertex of W is unmatched in M. Then we may construct a matching M' by removing from M all the edges belonging to $E(W) \cap M$, and adding all the remaining edges of W. This operation will be called *switching along path W*. Note that if W is of even length, then $|M'| = |M|$, while if W is of odd length and the last vertex of W is unmatched, then $|M'| = |M| + 1$. In the latter case we say that W is an *augmenting path for M*, since switching along W increases the size of the matching. The core observation of the classical algorithm for maximum matching in bipartite graphs is that a matching is not maximum if and only if there is an augmenting path for it.

A vertex cover of a graph is a set of vertices X such that every edge of the graph is incident to at least one vertex of X. Clearly, the size of a maximum matching is a lower bound for the size of a minimum vertex cover, as each vertex from the vertex cover can cover at most one edge of the matching. The classical Kőnig's theorem [41] states that for bipartite graphs equality holds: there exists a vertex cover of size $\mu(G)$, which is the minimum possible size. The set of arguments contained in this section in fact prove Kőnig's theorem.

The subset selector that will be used for the approximation of pathwidth is the following:

Definition 4.3. *By* matching selector $\mathfrak{M}$ *we denote a subset selector that assigns to every bipartite graph G the set of all the vertices of G that are matched in* **every** *maximum matching in G.*

Clearly, for any bipartite graph $G = (X, Y, E)$ we have that $|\mathfrak{M}(G) \cap X|, |\mathfrak{M}(G) \cap Y| \leq \mu(G)$, as any maximum matching of G is of size $\mu(G)$. It appears that $\mathfrak{M}$ is a symmetric and monotonic vertex cover selector. The symmetry is obvious. The crucial property of $\mathfrak{M}$ is monotonicity: its proof requires technical and careful analysis of alternating and augmenting paths in bipartite graphs. $\mathfrak{M}$ admits also an alternative characterization, expressed in Lemma 4.6, point (ii): it can be computed directly from any maximum matching by considering alternating paths originating in unmatched vertices. This observation can be utilized to construct an algorithm that maintains $\mathfrak{M}(G)$ efficiently during graph modifications. Moreover, from this alternative characterization it is clear that $\mathfrak{M}$ is a vertex cover selector. The following lemma expresses all the vital properties of $\mathfrak{M}$ that will be used in the approximation algorithm for pathwidth.

Lemma 4.4. $\mathfrak{M}$ *is a symmetric, monotonic vertex cover selector, which can be maintained together with a maximum matching of G with updates times $\mathcal{O}((\mu(G)+1)\cdot n)$ during vertex additions and deletions, where $n = |V(G)|$. Moreover, $|\mathfrak{M}(G) \cap X|, |\mathfrak{M}(G) \cap Y| \leq \mu(G)$ for every bipartite graph $G = (X, Y, E)$.*

We now proceed to the proof of Lemma 4.4. We split the proof into several lemmas. First, we show that $\mathfrak{M}$ is indeed monotonic.

Lemma 4.5. $\mathfrak{M}$ *is monotonic.*

Proof. Let $G' = G \setminus w$, where $G = (X, Y, E)$ is a bipartite graph and $w \in Y$.

Firstly, we prove that $\mathfrak{M}(G) \cap (Y \setminus \{w\}) \subseteq \mathfrak{M}(G') \cap Y$. For the sake of contradiction, assume that there is some $v \in \mathfrak{M}(G) \cap Y, v \neq w$, such that $v \notin \mathfrak{M}(G') \cap Y$. So there exists a maximum matching N of G' in which v is unmatched; we will also consider N as a (possibly not maximum) matching in G. Let M be any maximum matching in G. Since $v \in \mathfrak{M}(G)$, we have that v is matched in M. Construct a maximum path P in G that begins in v and alternates between matchings M and N (i.e., edges of P belong alternately to M and to N), starting with M. Note that every vertex of P that belongs to X is entered via an edge from M, and every vertex of P that belongs to Y is entered via an edge from N. As both w and v belong to Y and are not matched in N, this path does not enter w or v, hence it ends in some other vertex. Note also that P has length at least one as v is matched in M. Let $x \notin \{v, w\}$ be the last vertex of P. We consider two cases.

Assume first that $x \in X$, i.e., x is a vertex of the left side that is not matched in N. Then P is an augmenting path for N fully contained in G'. This contradicts the maximality of N.

Assume now that $x \in Y$, i.e., x is a vertex of the right side that is not matched in M. Then P is an alternating path for M in G and switching along P leaves v unmatched. This contradicts the fact that $v \in \mathfrak{M}(G)$.

Now we prove that $\mathfrak{M}(G) \cap X \supseteq \mathfrak{M}(G') \cap X$. We consider two cases.

Assume first that $w \in \mathfrak{M}(G)$. As w is matched in every maximum matching of G, after deleting w the size of the maximum matching drops by one: if there was a matching of the same size in G', it would constitute also a maximum matching in G that does not match w. Hence, if we take any maximum matching M of G and delete the edge incident to w, we obtain a maximum matching of G'. We infer that $\mathfrak{M}(G) \cap X \supseteq \mathfrak{M}(G') \cap X$, as every vertex of X that is unmatched in some maximum matching M in G, is also unmatched in some maximum matching G', namely in M with the edge incident to w removed.

Assume now that $w \notin \mathfrak{M}(G)$. Let M be any maximum matching of G in which w is unmatched. As M is also a matching in G', we infer that M is also a maximum matching in G' and the sizes of maximum matchings in G and G' are equal. Take any $v \in \mathfrak{M}(G') \cap X$ and for the sake of contradiction assume that $v \notin \mathfrak{M}(G)$. Let N be any maximum matching in G in which v is unmatched. Note that since $v \in \mathfrak{M}(G') \cap X$ and M is a maximum matching in G', then v is matched in M. Similarly as before, let us now construct a maximum path P in G that begins in v and alternates between matchings M and N, starting with M. Note that every vertex of P that belongs to X is entered via an edge from N, and every vertex of P that belongs to Y is entered via an edge from M. As $v \in X$, $w \in Y$, v is unmatched in N and w is unmatched in M, this path does not enter w or v, hence it ends in some other vertex. Note also that P has length at least one as v is matched in M. Let $x \notin \{v, w\}$ be the last vertex of P. Again, we consider two subcases.

In the first subcase we have $x \in X$, i.e., x is a vertex of the left side that is not matched in M. Then P is an alternating path for M in G' and switching along P leaves v unmatched. This contradicts the fact that $v \in \mathfrak{M}(G')$.

In the second subcase we have $x \in Y$, i.e., x is a vertex of the right side that is not matched in N. Then P is an augmenting path for N in G, which contradicts the maximality of N. $\square$

In order to prove that $\mathfrak{M}$ is a vertex cover selector and can be computed efficiently, we prove the following alternative characterization. The following lemma might be considered a folklore corollary of the classical matching theory, but for the sake of completeness we include its proof.

Lemma 4.6. *Let $G = (X, Y, E)$ be a bipartite graph and let M be a maximum matching in G. Let $A_0 = X \setminus V(M)$ be the set of unmatched vertices on the left side, and $B_0 = Y \setminus V(M)$ be the set of unmatched vertices on the right side. Moreover, let A be the set of vertices of X that can be reached via an alternating path from A_0, and symmetrically let B be the set of vertices of Y reachable by an alternating path from B_0. Then*

(i) there is no edge between A and B, and

(ii) $\mathfrak{M}(G) = V(G) \setminus (A \cup B)$.

Proof. We first prove point (i). For the sake of contradiction, assume that there is an edge between A and B, i.e., there is a vertex w in the set $N(A) \cap B$. Let v be a neighbor of w in A. We claim that w is reachable from A_0 via an alternating path. Assume otherwise, and take an alternating path P from A_0 reaching v. Observe that this path does not traverse w because then w would be reachable from A_0 via an alternating path. Moreover, $vw \notin M$, since if this was the case then vw would be used in P to access v. Hence, we can prolong P by the edge vw, thus reaching w by an alternation path from A_0, a contradiction. By the definition of B, w is also reachable from B_0 via an alternating path. The concatenation of these two paths is an alternating walk from A_0 to B_0, which contains an alternating simple subpath from A_0 to B_0. This subpath is an augmenting path for M, which contradicts maximality of M. Thus we infer that there is no edge between A and B, and point (i) is proven.

We now proceed to the proof that $\mathfrak{M}(G) = V(G) \setminus (A \cup B)$. On one hand, every vertex v belonging to A or B is not matched in some maximum matching: we just modify M by switching along the alternating path connecting a vertex unmatched in M with v, obtaining another maximum matching in which v is unmatched. Hence, $\mathfrak{M}(G) \subseteq V(G) \setminus (A \cup B)$. We are left with proving that $\mathfrak{M}(G) \supseteq V(G) \setminus (A \cup B)$.

Consider the set A and the set $N(A)$. We already know that $N(A)$ is disjoint with B, so also from B_0. Hence, every vertex of $N(A)$ must be matched in M. Moreover, we have that every vertex of $N(A)$ must be in fact matched to a vertex of A, as the vertices matched to $N(A)$ are also reachable via alternating walks from A_0. Similarly, every vertex of $N(B)$ is matched to a vertex of B.

We now claim that $|X \setminus A| + |N(A)| = |M|$. As $A_0 \subseteq A$, every vertex of $X \setminus A$ as well as every vertex of $N(A)$ is matched in M. Moreover, as there is no edge between A and $Y \setminus N(A)$, every edge of M has an endpoint either in $X \setminus A$ or in $N(A)$. It remains to show that no edge of M can have one endpoint in $X \setminus A$ and second in $N(A)$; this, however follows from the fact that vertices of $N(A)$ are matched to vertices of A, proved in the previous paragraph. Similarly we have that $|Y \setminus B| + |N(B)| = |M|$.

It follows that sets $(X \setminus A) \cup N(A)$ and $N(B) \cup (X \setminus B)$ are vertex covers of G of size $|M|$. Note that every vertex cover C of G of size $|M|$ must be fully matched by any matching N of size $|M|$, as every edge of N is incident to at least one vertex from C. Hence, every vertex of both $(X \setminus A) \cup N(A)$ and of $N(B) \cup (X \setminus B)$ is matched in every maximum matching of G. Since $N(A) \subseteq Y \setminus B$ and $N(B) \subseteq X \setminus A$, we have that $(X \setminus A) \cup N(A) \cup N(B) \cup (Y \setminus B) = V(G) \setminus (A \cup B)$. Thus $\mathfrak{M}(G) \supseteq V(G) \setminus (A \cup B)$. $\square$

From now on we use the terminology introduced in Lemma 4.6. Note that Lemma 4.6 implies that sets A, B do not depend on the choice of matching M, but only on graph G. Also, point (i) of Lemma 4.6 shows that $\mathfrak{M}$ is a vertex cover selector. We now present an incremental algorithm that maintains $\mathfrak{M}(G)$ together with a maximum matching of G during vertex additions and deletions.

Lemma 4.7. *There exists an algorithm that maintains a maximum matching M and $\mathfrak{M}(G)$ for a bipartite graph $G = (X, Y, E)$ during vertex addition and deletion operations. The update time is $\mathcal{O}((\mu(G) + 1) \cdot n)$, where $n = |V(G)|$.*

Proof. Observe that, by Lemma 4.6, $\mathfrak{M}(G)$ can be computed from M in $\mathcal{O}(|G|)$ time: we simply compute sets A_0, B_0 and apply breadth-first search from the whole A_0 to compute A, and breadth-first search from the whole B_0 to compute B. Both of these searches take $\mathcal{O}(|G|)$ time; note that by Kőnig's theorem a bipartite graph G on n vertices can have at most $\mathcal{O}(\mu(G) \cdot n)$ edges, so $|G| = \mathcal{O}((\mu(G) + 1) \cdot n)$. Hence, we just need to maintain a maximum matching.

During vertex addition, the size of the maximum matching can increase by at most 1, so we may simply add the new vertex and run one iteration of the standard breadth-first search procedure checking, whether there is an augmenting path in the new graph; this takes $\mathcal{O}(|G|)$ time. If this is the case, we modify the matching along this path and we know that we obtained a maximum matching in the new graph. Otherwise, the size of the maximum matching does not increase, so we do not need to modify the current one. Similarly, during vertex deletion we simply delete the vertex together with possibly at most one edge of the matching incident to it. The size of the stored matching might have decreased by 1; in this case again we run one iteration of checking whether there is an augmenting path, again in $\mathcal{O}(|G|)$ time. If this is the case, we augment the matching using this path, obtaining a new matching about which we know that it is maximum. Otherwise, we know that the current matching is maximum, so we do not need to modify it. □

Lemmas 4.5, 4.6 and 4.7 together with previous observations prove Lemma 4.4.

Let us conclude with the following remark. Instead of selector $\mathfrak{M}$ one could introduce a selector $\mathfrak{M}'$ defined as follows in terms of Lemma 4.6: $\mathfrak{M}'((X, Y, E)) = (X \setminus A) \cup N(A)$. The proof of Lemma 4.6 shows that this definition does not depend on the choice of maximum matching M, and moreover that selector $\mathfrak{M}'$ is in fact a vertex cover selector of size exactly $\mu(G)$. Obviously, $\mathfrak{M}'$ can be maintained in the same manner as $\mathfrak{M}$ during vertex additions and deletions, so the only missing piece is showing that both $\mathfrak{M}'$ and $\mathfrak{M}'^{\mathrm{rev}}$ are monotonic; note here that since $\mathfrak{M}$ is symmetric, when working with $\mathfrak{M}$ we needed to perform just one such check. This claim appears to be true; however, the proof is significantly longer and more technical than the proof of Lemma 4.5. In addition, even though selector $\mathfrak{M}'$ has better guarantees on the size than $\mathfrak{M}$, unfortunately this gain appears to be not useful at the point when we apply $\mathfrak{M}$ in the approximation algorithm for pathwidth. In other words, replacing $\mathfrak{M}$ with $\mathfrak{M}'$ does not result in better approximation ratio, even though it may be shown that when $\mathfrak{M}'$ is applied instead of $\mathfrak{M}$, we have a better guarantee of $5k$ instead of $6k$ on the sizes of intersections of each two consecutive bags (so-called *adhesions*). Therefore, for the sake of simpler and more concise arguments we have chosen to include analysis using selector $\mathfrak{M}$ instead of $\mathfrak{M}'$.

4.1.2 The Buss selector

In this subsection we introduce the subset selector that will be used in the exact algorithm for pathwidth. This selector is inspired by the classical kernelization algorithm for the Vertex Cover problem of Buss [8].

Definition 4.8. *Let $G = (X, Y, E)$ be a bipartite graph and ℓ be a nonnegative integer. A vertex v is called* ℓ-important *if $d(v) > \ell$, and* ℓ-unimportant *otherwise. A* Buss selector *is a subset selector $\mathfrak{B}_\ell$ that returns all vertices of X that are either ℓ-important, or have at least one ℓ-unimportant neighbor.*

Note that Buss selector is highly non-symmetric, as it chooses vertices only from the left side. Moreover, it is not necessarily a vertex cover selector. However, both $\mathfrak{B}_\ell$ and $\mathfrak{B}_\ell^{\mathrm{rev}}$ behave in a nice manner.

Lemma 4.9. *Both $\mathfrak{B}_\ell$ and $\mathfrak{B}_\ell^{\mathrm{rev}}$ are monotonic.*

Proof. Monotonicity of $\mathfrak{B}_\ell$ is equivalent to the observation that if $v \in X$ is ℓ-unimportant and has only ℓ-important neighbors, then deletion of any vertex of Y cannot make v ℓ-important or create ℓ-unimportant neighbors of v. This holds because the degrees of surviving neighbors of v do not change.

Monotonicity of $\mathfrak{B}_\ell^{\mathrm{rev}}$ is equivalent to the observation that if $w \in Y$ is chosen by $\mathfrak{B}_\ell^{\mathrm{rev}}$ because it is ℓ-important, then after deletion of any other $w' \in Y$ its degree does not change so it stays ℓ-important, and if it had an ℓ-unimportant neighbor, then after deletion of any other $w' \in Y$ this neighbor will still be ℓ-unimportant. $\square$

We now prove that $\mathfrak{B}_\ell$ does not choose too many vertices unless the bipartite graph G contains a large matching.

Lemma 4.10. *If $|\mathfrak{B}_\ell(G)| > \ell^2 + \ell$, then G contains a matching of size $\ell + 1$.*

Proof. As $|\mathfrak{B}_\ell(G)| > \ell^2+\ell$, in $\mathfrak{B}_\ell(G)$ there are at least $\ell+1$ ℓ-important vertices, or at least ℓ^2+1 vertices with an ℓ-unimportant neighbor. In both cases we construct the matching greedily.

In the first case we iteratively take an ℓ-important vertex of $\mathfrak{B}_\ell(G)$ and match it with any its neighbor that is not matched so far. As there is at least $\ell+1$ of these neighbors and at most ℓ were used so far, we can always find one not matched so far.

In the second case we take an arbitrary vertex v_1 of $\mathfrak{B}_\ell(G)$ that has an ℓ-unimportant neighbor, and find any its ℓ-unimportant neighbor w_1. We add v_1w_1 to the constructed matching and mark all the at most ℓ neighbors of w_1 as used. Then we take an arbitrary unused vertex v_2 of $\mathfrak{B}_\ell(G)$ that has an ℓ-unimportant neighbor, find any its ℓ-unimportant neighbor w_2 (note that $w_2 \neq w_1$ since v_2 was not marked), add v_2w_2 to the constructed matching and mark all the at most ℓ neighbors of w_2 as used. We continue in this manner up to the point when a matching of size $\ell + 1$ is constructed. Note that there will always be an unmarked vertex of $\mathfrak{B}_\ell(G)$ with an ℓ-unimportant neighbor, as at the beginning there are at least $\ell^2 + 1$ of them and after i iterations at most $i \cdot \ell$ are marked as used. $\square$

We prove that $\mathfrak{B}_\ell$ can be also evaluated efficiently.

Lemma 4.11. *There exists an algorithm which maintains $\mathfrak{B}_\ell(G)$ for a bipartite graph $G = (X, Y, E)$ during operations of vertex addition and vertex deletion with update times $\mathcal{O}(\ell n)$, where $n = |V(G)|$.*

Proof. With every vertex of X we maintain its degree and a counter of ℓ-unimportant neighbors. We also maintain degrees of vertices of Y. The degree gives us information whether the vertex is ℓ-important.

Let us first examine adding vertex v to the left side. We need to increase the degrees of neighbors of v, so some of them may become ℓ-important. For every such vertex that became ℓ-important — note that its degree is exactly $\ell+1$ — we examine its $\ell+1$ neighbors and decrement their ℓ-unimportant neighbors' counters. If it drops to zero, we delete this vertex from $\mathfrak{B}_\ell(G)$

unless it's ℓ-important. Finally, we count how many neighbors of v are ℓ-unimportant, set the counter appropriately and add v to $\mathfrak{B}_\ell(G)$ if necessary.

Let us now examine adding vertex w to the right side. We need to increase the degrees of neighbors of w, so some of them may become ℓ-important and thus chosen by $\mathfrak{B}_\ell(G)$. Moreover, if w is ℓ-unimportant, then we increment the ℓ-unimportant neighbors' counters of neighbors of w; if it is incremented from 0 to 1, then the vertex becomes chosen by $\mathfrak{B}_\ell(G)$, assuming that it was not already ℓ-important.

Now we examine deleting vertex v from the left side. We iterate through the neighbors of v and decrement their degrees. If any of them ceases to be ℓ-important, we iterate through all its ℓ neighbors and increment the counters of ℓ-unimportant neighbors. If for some of these neighbors the counter was incremented from 0 to 1, the neighbor becomes chosen by $\mathfrak{B}_\ell(G)$, assuming that it was not already ℓ-important.

Finally, we examine deleting vertex w from the right side. Firstly, we check if w was ℓ-important. Then we iterate through neighbors of w decreasing the degree and ℓ-unimportant neighbors' counters, if necessary. If any of the neighbors ceases to be ℓ-important or have an ℓ-unimportant neighbor, it becomes not chosen to $\mathfrak{B}_\ell(G)$. $\square$

4.2 The algorithms

In this subsection we present the algorithms for computing pathwidth. We begin with the approximation algorithm and then proceed to the exact algorithm. We introduce the approximation algorithm with an additional parameter ℓ; taking $\ell = 4k$ gives the promised 6-approximation, but as we will seelater, modifying ℓ may be useful to improve the quality of obtained degree tangle.

Theorem 4.12. *There exists an algorithm that given a semi-complete digraph T on n vertices and integers k and $\ell \geq 4k$, in time $\mathcal{O}(kn^2)$ outputs one of the following:*

- *an $(\ell+2,k)$-degree tangle in T;*
- *a $(k+1,k)$-matching tangle in T;*
- *a path decomposition of T of width at most $\ell+2k$.*

In the first two cases the algorithm can correctly conclude that $\mathbf{pw}(T) > k$.

Proof. The last sentence follows from Lemmas 3.8 and 3.11. We proceed to the algorithm.

The algorithm first computes any outdegree ordering $\sigma = (v_1, v_2, \ldots, v_n)$ of $V(T)$ in $\mathcal{O}(n^2)$ time. Then in $\mathcal{O}(n)$ time we check if there is an index i such that $d^+(v_{i+\ell+1}) \leq d^+(v_i) + k$. If this is true, then $\{v_i, v_{i+1}, \ldots, v_{i+\ell+1}\}$ is an $(\ell+2,k)$-degree tangle which can be safely output by the algorithm. From now on we assume that such a situation does not occur, i.e, $d^+(v_{i+\ell+1}) > d^+(v_i) + k$ for every index i.

Let separation sequence $R_0 = ((A_0,B_0),(A_1,B_1),\ldots,(A_{n-\ell},B_{n-\ell}))$ be defined as follows. Let us define $S_i^0 = \{v_{i+1}, v_{i+2}, \ldots, v_{i+\ell}\}$ and let $H_i = (X_i, Y_i, E_i)$ be a bipartite graph, where $X_i = \{v_1, \ldots, v_i\}$, $Y_i = \{v_{i+\ell+1}, v_{i+\ell+2}, \ldots, v_n\}$ and $xy \in E_i$ if and only if $(x,y) \in E(T)$. If $\mu(H_i) > k$, then vertices matched in a maximum matching of H_i form a $(k+1,k)$-matching tangle in T, which can be safely output by the algorithm. Otherwise, let $S_i = S_i^0 \cup \mathfrak{M}(H_i)$ and we set $A_i = X_i \cup S_i$ and $B_i = Y_i \cup S_i$; the fact that (A_i,B_i) is a separation follows from the fact that $\mathfrak{M}$ is a vertex cover selector. Finally, we add separations $(\emptyset, V(T))$ and $(V(T),\emptyset)$ at the ends of the sequence, thus obtaining separation sequence R. We claim that R is a separation chain. Note that if we prove it, by Lemma 2.12 the width of the corresponding path decomposition is upper bounded by $\max_{0\leq i\leq n-\ell-1} |\{v_{i+1}, v_{i+2}, \ldots, v_{i+\ell+1}\} \cup (\mathfrak{M}(H_i) \cap X_i) \cup (\mathfrak{M}(H_{i+1}) \cap Y_{i+1})| - 1 \leq \ell + 1 + 2k - 1 = \ell + 2k$, by monotonicity of $\mathfrak{M}$ (Lemma 4.5).

It suffices to show that for every i we have that $A_i \subseteq A_{i+1}$ and $B_i \supseteq B_{i+1}$. This, however, follows from Lemma 4.2 and the fact that $\mathfrak{M}$ is symmetric and monotonic. H_{i+1} differs from H_i by deletion of one vertex on the right side and addition of one vertex on the left side, so we have that A_{i+1} differs from A_i only by possibly incorporating vertex $v_{i+\ell+1}$ and some vertices from Y_{i+1} that became chosen by $\mathfrak{M}$, and B_{i+1} differs from B_i only by possibly losing vertex v_{i+1} and some vertices from X_i that ceased to be chosen by $\mathfrak{M}$.

Separation chain R can be computed in $\mathcal{O}(kn^2)$ time: we consider consecutive sets S_i^0 and maintain the graph H_i together with a maximum matching in it and $\mathfrak{M}(H_i)$. As going to the next set S_i^0 can be modeled by one vertex deletion and one vertex additions in graph H_i, by Lemma 4.4 we have that the time needed for an update is $\mathcal{O}(kn)$; note that whenever the size of the maximum matching exceeds k, we terminate the algorithm by outputting the obtained matching tangle. As we make $\mathcal{O}(n)$ updates, the time bound follows. Translating a separation chain into a path decomposition can be done in $\mathcal{O}(\ell n)$ time, since we can store the separators along with the separations when constructing them. $\square$

We now present the exact algorithm for pathwidth.

Theorem 4.13. *There exists an algorithm that, given a semi-complete digraph T on n vertices and an integer k, in $2^{\mathcal{O}(k\log k)} \cdot n^2$ time computes a path decomposition of T of width at most k, or correctly concludes that no such exists.*

Proof. We say that a separation chain R in T is *feasible* if it has width at most k. By the remarks after Lemma 2.12, instead of looking for a path decomposition of width k we may look for a feasible separation chain. Transformation of this separation chain into a path decomposition can be carried out in $\mathcal{O}(kn)$ time, assuming that we store the separators along with the separations.

The opening step of the algorithm is fixing some outdegree ordering $\sigma = (v_1, v_2, \ldots, v_n)$ of $V(T)$. For $i = 0, 1, \ldots, n$, let H_i be a bipartite graph with left side $X_i = \{v_1, v_2, \ldots, v_i\}$ and right $Y_i = \{v_{i+1}, v_{i+2}, \ldots, v_n\}$, where $xy \in E(H_i)$ if $(x, y) \in E(T)$. As in the proof of Theorem 4.12, in $\mathcal{O}(n)$ time we check if there is an index i such that $d^+(v_{i+4k+1}) \leq d^+(v_i) + k$. If this is true, then $\{v_i, v_{i+1}, \ldots, v_{i+4k+1}\}$ is a $(4k+2, k)$-degree tangle and the algorithm may safely provide a negative answer by Lemma 3.8. From now on we assume that such a situation does not occur.

We define a subclass of *trimmed* separation chains. Every feasible separation chain can be adjusted to a trimmed separation chain by deleting some vertices from sets A_i, B_i; note that the new separation chain created in such a manner will also be feasible, as bags of the corresponding path decomposition can only get smaller. Hence, we may safely look for a separation chain that is trimmed.

Before proceeding with formal arguments, let us explain some intuition behind trimmed separation chains. A priori, a bag of a path decomposition of T of width at most k can contain some redundant vertices that do not really contribute to the separation property (Property (iii) of Definition 2.9). For instance, if T admits a decomposition of width $k/2$, then one could pick an arbitrary set of $k/2$ vertices of T and incorporate them in every bag; this gives us roughly $n^{k/2}$ possible decompositions, which is too large a number for an FPT algorithm. Hence, we need to restrict our attention to path decompositions that are cleaned from redundant parts of bags; these decompositions are precisely the ones that correspond to trimmed separation chains. The Buss selector $\mathfrak{B}_\ell$ will be the tool responsible for identifying which part of the bags are redundant and can be removed.

We proceed to the definition of a trimmed separation chain. We fix $m = 5k + 1$. Let (A, B) be any separation in T, and let α, β be any indices between 0 and n such that $\alpha = \max(0, \beta -$

$(4k+1))$. We say that (A,B) is *trimmed with respect to* (α,β) if (i) $Y_\beta \subseteq B \subseteq Y_\alpha \cup \mathfrak{B}_m(H_\alpha)$ and (ii) $X_\alpha \subseteq A \subseteq X_\beta \cup \mathfrak{B}_m^{\mathrm{rev}}(H_\beta)$. (A,B) is *trimmed* if it is trimmed with respect to any such pair (α,β). A separation chain is *trimmed* if every its separation is trimmed.

For a separation (A,B) let us define the *canonical index* $\beta = \beta((A,B))$ as the only integer between 0 and n such that $d^+(v_j) < |A|$ for $j \leq \beta$ and $d^+(v_j) \geq |A|$ for $j > \beta$. Similarly, the *canonical index* $\alpha = \alpha((A,B))$ is defined as $\alpha((A,B)) = \max(0, \beta((A,B)) - (4k+1))$. Observe that by the assumed properties of ordering σ we have that vertices in X_α have outdegrees smaller than $|A| - k$.

Firstly, we observe that if (A,B) is a separation and α,β are its canonical indices, then $X_\alpha \subseteq A$ and $Y_\beta \subseteq B$. This follows from the fact that vertices in $A \setminus B$ have outdegrees smaller than $|A|$, hence they cannot be contained in Y_β, while vertices in $B \setminus A$ have outdegrees at least $|A \setminus B| \geq |A| - k$, hence they cannot be contained in X_α. Concluding, vertices of X_α may belong only to $A \setminus B$ or $A \cap B$ (left side or the separator), vertices of Y_β may belong only to $B \setminus A$ or $A \cap B$ (right side or the separator), while for vertices of the remaining part $Y_\alpha \cap X_\beta$ neither of the three possibilities is excluded.

We now show how to transform any separation chain into a trimmed one. Take any separation chain R that contains only separations of order at most k, and obtain a sequence of separations R' as follows. We take every separation (A,B) from R; let α,β be its canonical indices. We delete $X_\alpha \setminus \mathfrak{B}_m(H_\alpha)$ from B and $Y_\beta \setminus \mathfrak{B}_m^{\mathrm{rev}}(H_\beta)$ from A, thus obtaining a new pair (A',B') that is inserted into R' in place of (A,B).

Claim 1. *R' is a trimmed separation chain.*

Proof. We need to prove that every such pair (A',B') is a separation, and that all these separations form a separation chain. The fact that such a separation chain is trimmed follows directly from the definition of the performed operation and the observations on canonical indices.

First, we check that $A' \cup B' = V(T)$. This follows from the fact from A we remove only vertices of Y_β while from B we remove only vertices from X_α, but after the removal X_α is still covered by A' and Y_β by B'.

Now we check that $E(A' \setminus B', B' \setminus A') = \emptyset$. Assume otherwise, that there is a pair $(v,w) \in E(T)$ such that $v \in A' \setminus B'$ and $w \in B' \setminus A'$. By the construction of (A',B') and the fact that (A,B) was a separation we infer that either $v \in X_\alpha \setminus \mathfrak{B}_m(H_\alpha)$ or $w \in Y_\beta \setminus \mathfrak{B}_m^{\mathrm{rev}}(H_\beta)$. We consider the first case, as the second is symmetrical.

Since $w \notin A'$ and $X_\alpha \subseteq A'$, we have that $w \in Y_\alpha$, so vw is an edge in H_α. As $v \notin \mathfrak{B}_m(H_\alpha)$, we have that in H_α vertex v is m-unimportant and has only m-important neighbors. Hence w is m-important in H_α. Observe now that w cannot be contained in $B \setminus A$, as there is more than $m > k$ vertices in X_α being tails of arcs directed toward w, and only k of them can be in separator $A \cap B$ leaving at least one belonging to $A \setminus B$ (recall that vertices from X_α cannot belong to $B \setminus A$). Hence $w \in A$. As $w \notin A'$, we have that $w \in Y_\beta \setminus \mathfrak{B}_m^{\mathrm{rev}}(H_\beta)$. However, w was an m-important vertex on the right side of H_α, so as it is also on the right side of H_β, it is also m-important in H_β. This is a contradiction with $w \notin \mathfrak{B}_m^{\mathrm{rev}}(H_\beta)$.

We conclude that (A',B') is indeed a separation.

Finally, we check that R' is a separation chain. Consider two separations (A_1,B_1), (A_2,B_2) in R, such that $A_1 \subseteq A_2$ and $B_1 \supseteq B_2$. Let $\alpha_1,\beta_1,\alpha_2,\beta_2$ be canonical indices of (A_1,B_1) and (A_2,B_2), respectively. It follows that $\alpha_1 \leq \alpha_2$ and $\beta_1 \leq \beta_2$. Hence, graph H_{α_2} can be obtained from H_{α_1} via a sequence of vertex deletions on the right side and vertex additions on the left side. As $\mathfrak{B}_m$ and $\mathfrak{B}_m^{\mathrm{rev}}$ are monotonic (Lemma 4.9), by Lemma 4.2 we have that every vertex deleted from B_1 while constructing B_1' is also deleted from B_2 while constructing B_2' (assuming it belongs to B_2). Hence, $B_2' \subseteq B_1'$. A symmetric argument shows that $A_2' \supseteq A_1'$. ⌟

We proceed to the algorithm itself. As we have argued, we may look for a trimmed feasible separation chain. Indeed if T admits a path decomposition of width at most k, then by Lemma 2.12 it admits a feasible separation chain, which by Claim 1 can be turned into a trimmed feasible separation chain. On the other hand, if T admits a trimmed feasible separation chain, then this separation chain can be turned into a path decomposition of T of width at most k using Lemma 2.12.

Let $\mathcal{N}$ be the family of trimmed separations of T of order at most k. We construct an auxiliary digraph D with vertex set $\mathcal{N}$ by putting an arc $((A,B),(A',B')) \in E(D)$ if and only if $A \subseteq A'$, $B \supseteq B'$ and $|A' \cap B| \leq k+1$. Then paths in D from $(\emptyset, V(T))$ to $(V(T), \emptyset)$ correspond to feasible trimmed separation chains.

We prove that either the algorithm can find an obstacle for admitting path decomposition of width at most k, or D has size at most $2^{\mathcal{O}(k \log k)} \cdot n$ and can be constructed in time $2^{\mathcal{O}(k \log k)} \cdot n^2$. Hence, any linear-time reachability algorithm in D runs within claimed time complexity bound.

Consider any indices α, β such that $\alpha = \max(0, \beta - (4k+1))$. Observe that if $|\mathfrak{B}_m(H_\alpha)| > m^2+m$, by Lemma 4.10 we can find a matching of size $m+1$ in H_α. At most $4k+1 = m-k$ edges of this matching have the right endpoint in $Y_\alpha \cap X_\beta$, which leaves us at least $k+1$ edges between X_α and Y_β. Such a structure is a $(k+1,k)$-matching tangle in T, so by Lemma 3.11 the algorithm may provide a negative answer. A symmetrical reasoning shows that if $|\mathfrak{B}_m^{\mathrm{rev}}(H_\beta)| > m^2+m$, then the algorithm can also provide a negative answer.

If we assume that these situations do not occur, we can characterize every trimmed separation (A,B) of order at most k by:

- a number β, where $0 \leq \beta \leq n$;
- a mask on the vertices from $Y_\alpha \cap X_\beta$, denoting for each of them whether it belongs to $A \setminus B$, $A \cap B$ or to $B \setminus A$ (at most 3^{4k+1} options);
- subsets of size at most k of $X_\alpha \cap \mathfrak{B}_m(H_\alpha)$ and $Y_\beta \cap \mathfrak{B}_m^{\mathrm{rev}}(H_\beta)$, denoting which vertices belong to $A \cap B$ (at most $\left(k\binom{\mathcal{O}(k^2)}{k}\right)^2 = 2^{\mathcal{O}(k \log k)}$ options).

Moreover, if (A',B') is an outneighbor of (A,B) in D, then it must have parameter β' not larger than $\beta + (5k+2)$, as otherwise we have a guarantee that $|A' \cap B| \geq |X_{\alpha'} \cap Y_\beta| \geq k+2$, and also not smaller than $\beta - (5k+2)$, as otherwise we have a guarantee that $|A| \geq |X_\alpha| > |A'|$. Hence, the outdegrees in D are bounded by $2^{\mathcal{O}(k \log k)}$.

This gives raise to the following algorithm constructing D in time $2^{\mathcal{O}(k \log k)} \cdot n^2$.

- First, we enumerate $\mathcal{N}$. We scan through the order σ with an index β maintaining graphs H_α, H_β for $\alpha = \max(0, \beta - (4k+1))$, along with $\mathfrak{B}_m(H_\alpha)$ and $\mathfrak{B}_m^{\mathrm{rev}}(H_\beta)$. Whenever cardinality of any of these sets exceeds m^2+m, we terminate the algorithm providing a negative answer. By Lemma 4.11 we can bound the update time by $\mathcal{O}(kn)$. For given index β, we list all $2^{\mathcal{O}(k \log k)}$ pairs (A,B) having this particular β in characterization from the previous paragraph. For every such pair, in $\mathcal{O}(kn)$ we check whether it induces a separation of order k, by testing emptiness of set $E(A \setminus B, B \setminus A)$ using at most $\mathcal{O}(k)$ operations of vertex deletion/addition on the graph H_α. We discard all the pairs that do not form such a separation; all the remaining ones are exactly separations that are trimmed with respect to (α, β).
- For every separation (A,B) characterized by parameter β, we check for all the $2^{\mathcal{O}(k \log k)}$ separations (A',B') with parameter β' between $\beta - (5k+2)$ and $\beta + (5k+2)$, whether we should put an arc from (A,B) to (A',B'). Each such a check can be performed in $\mathcal{O}(k)$ time, assuming that we store the separator along with the separation.

Having constructed D, we run a linear-time reachability algorithm to check whether $(V(T), \emptyset)$ can be reached from $(\emptyset, V(T))$. If not, we provide a negative answer; otherwise, the path corresponds to a feasible separation chain which can be transformed into a path decomposition in $\mathcal{O}(kn)$ time. $\square$

5 Algorithms for computing cutwidth and for other ordering problems

In this section we present the algorithms for computing cutwidth and related problems. We start with the approximation algorithm for cutwidth, which follows immediately from the results of Section 3. Then we proceed to the description of the exact algorithm for cutwidth that runs in time $2^{\mathcal{O}(\sqrt{k \log k})} \cdot n^2$. It appears that using our approach it is possible to give a subexponential parameterized algorithm not only for cutwidth, but also for two related problems, namely Feedback Arc Set and Optimal Linear Arrangement.

5.1 Approximation of cutwidth

The results of Section 3 immediately yield the following theorem.

Theorem 5.1. *Let T be a semi-complete digraph and let $m(t) = 64t^2 + 18t + 1$. Then any outdegree ordering of $V(T)$ has width at most $m(\mathbf{ctw}(T))$.*

Proof. Let σ be any outdegree ordering of $V(T)$. If σ had width more than $m(\mathbf{ctw}(T))$, then one of the partitions $(\sigma[\alpha], V(T) \setminus \sigma[\alpha])$ would be a $(m(\mathbf{ctw}(T))+1)$-backward tangle. Existence of such a structure is a contradiction with Lemma 3.14. $\square$

This gives raise to a straightforward approximation algorithm for cutwidth of a semi-complete digraph that simply sorts the vertices with respect to outdegrees, and then scans through the ordering checking whether it has small width. Note that this scan may be performed in $\mathcal{O}(|V(T)|^2)$ time, as we maintain the cutset between the prefix and the suffix of the ordering by iteratively moving one vertex from the suffix to the prefix.

Theorem 5.2. *There exists an algorithm which, given a semi-complete digraph T on n vertices and an integer k, in time $\mathcal{O}(n^2)$ outputs an ordering of $V(T)$ of width at most $m(k)$ or a $(m(k)+1)$-backward tangle in T, where $m(t) = 64t^2 + 18t + 1$. In the second case the algorithm concludes that $\mathbf{ctw}(T) > k$.*

5.2 Additional problem definitions

We now introduce the formal definitions of problems Feedback Arc Set and Optimal Linear Arrangement, and prove their basic properties that will be useful in the algorithms.

Feedback Arc Set

Definition 5.3. *Let T be a digraph. A subset $F \subseteq E(T)$ is called a* feedback arc set *if $T \setminus F$ is acyclic.*

The Feedback Arc Set problem in semi-complete digraphs is then defined as follows.

Feedback Arc Set (FAS) in semi-complete digraphs

Input: A semi-complete digraph T and a nonnegative integer k.
Parameter: k
Question: Is there a feedback arc set of T of size at most k?

We have the following easy observation that enables us to view FAS as a graph layout problem.

Lemma 5.4. *Let T be a digraph. Then T admits a feedback arc set of size at most k if and only if there exists an ordering $(v_1, v_2, \ldots, v_n)$ of $V(T)$ such that at most k arcs of $E(T)$ are directed forward in this ordering, i.e., are of form (v_i, v_j) for $i < j$.*

Proof. If F is a feedback arc set in T then the ordering can be obtained by taking any reverse topological ordering of $T \setminus F$. On the other hand, given the ordering we may simply define F to be the set of forward edges. $\square$

Optimal Linear Arrangement

Definition 5.5. *Let T be a digraph and $(v_1, v_2, \ldots, v_n)$ be an ordering of its vertices. Then the* cost *of this ordering is defined as*

$$\sum_{(v_i,v_j)\in E(T)} (j-i)\cdot[j>i],$$

that is, every arc directed forwards in the ordering contributes to the cost with the distance between the endpoints in the ordering.

Whenever the ordering is clear from the context, we also refer to the contribution of a given arc to its cost as to the *length* of this arc. By a simple reordering of the computation we obtain the following:

Lemma 5.6. *For a digraph T and ordering $(v_1, v_2, \ldots, v_n)$ of $V(T)$, the cost of this ordering is equal to:*

$$\sum_{t=1}^{n-1} |E(\{v_1, v_2, \ldots, v_t\}, \{v_{t+1}, v_{t+2}, \ldots, v_n\})|.$$

Proof. Observe that

$$\begin{aligned}\sum_{(v_i,v_j)\in E(T)} (i-j)\cdot[i>j] &= \sum_{(v_i,v_j)\in E(T)} \sum_{t=1}^{n-1}[i\le t<j] \\ &= \sum_{t=1}^{n-1} \sum_{(v_i,v_j)\in E(T)} [i\le t<j] \\ &= \sum_{t=1}^{n-1} |E(\{v_1, \ldots, v_t\}, \{v_{t+1}, \ldots, v_n\})|.\end{aligned}$$

$\square$

The problem Optimal Linear Arrangement in semi-complete digraphs is then defined as follows.

Optimal Linear Arrangement (OLA) in semi-complete digraphs	
Input:	A semi-complete digraph T and a nonnegative integer k.
Parameter:	k
Question:	Is there an ordering of $V(T)$ of cost at most k?

5.3 k-cuts of semi-complete digraphs

In this section we introduce the main technical ingredient of our algorithms, namely the concept of a *k-cut*.

Definition 5.7. *A k-*cut *of a multidigraph T is a partition (X, Y) of $V(T)$ with the following property: there are at most k arcs $(u, v) \in E(T)$ such that $u \in X$ and $v \in Y$. For a multidigraph T, by $\mathcal{N}(T, k)$ we denote the family of all the k-cuts of T.*

Intuitively, the k-cuts will form the state space of our dynamic programs, so it is essential to prove that their number is small when the given semi-complete digraph has small cutwidth or feedback vertex set number. We first provide some results on efficient enumeration of k-cuts, and then prove that when cutwidth or feedback vertex set number is at most k, then the number of k-cuts is bounded subexponentially in k.

5.3.1 Enumerating k-cuts

The following lemma shows that k-cuts can be enumerated efficiently.

Lemma 5.8. *Let D be a multidigraph and let X_0, Y_0 be disjoint sets of vertices of D. Then the family of all the k-cuts (X, Y) such that $X_0 \subseteq X$ and $Y_0 \subseteq Y$ can be enumerated with polynomial-time delay, where each k-cut is enumerated together with number $|E(X, Y)|$.*

Proof. Let $\sigma = (v_1, v_2, \ldots, v_p)$ be an arbitrary ordering of vertices of $V(D) \setminus (X_0 \cup Y_0)$. We perform a classical branching strategy. We start with $X = X_0$ and $Y = Y_0$, and consider the vertices in order σ, at each step branching into one of the two possibilities: vertex v_i is to be incorporated into X or into Y. However, after assigning each consecutive vertex we run a max-flow algorithm from X to Y to find the size of a minimum edge cut between X and Y. If this size is more than k, we terminate the branch as we know that it cannot result in any solutions found. Otherwise we proceed. We output a partition after the last vertex, v_n, is assigned a side; note that the last max-flow check ensures that the output partition is actually a k-cut, and finds the output size of the cut as well. Moreover, as during the algorithm we consider only branches that can produce at least one k-cut, the next partition will be always found within polynomial waiting time, proportional to the depth of the branching tree times the time needed for computations at each node of the branching tree. □

Setting $X_0 = Y_0 = \emptyset$ gives an algorithm enumerating k-cuts of D with polynomial-time delay. The running time of Lemma 5.8 is unfortunately not satisfactory if one would like to design a linear-time algorithm. Therefore, we prove that k-cuts of a semi-complete digraph of small cutwidth can be enumerated more efficiently.

Lemma 5.9. *There exists an algorithm that, given a semi-complete digraph T on n vertices together with nonnegative integers k and B, works in $\mathcal{O}(n^2 + B \cdot k^{\mathcal{O}(1)} \cdot n)$, and either:*

- *correctly concludes that* $\mathbf{ctw}(T) > k$*;*
- *correctly concludes that the number of* k*-cuts of* T *is more than* B*;*
- *or outputs the whole family* $\mathcal{N}(T,k)$ *of* k*-cuts of* T *with a guarantee that* $|\mathcal{N}(T,k)| \le B$*, where each* k*-cut* (X,Y) *is output together with* $|E(X,Y)|$*.*

Proof. If $n \le 8k+1$, we run the enumeration algorithm of Lemma 5.8 and terminate it if the number of enumerated k-cuts exceeds B. Since k-cuts are enumerated with polynomial delay and $n = \mathcal{O}(k)$, the algorithm works in $\mathcal{O}(B \cdot k^{\mathcal{O}(1)})$ time.

Assume then that $n > 8k+1$. First, the algorithm computes in $\mathcal{O}(n^2)$ time any outdegree ordering $\sigma = (v_1, v_2, \ldots, v_n)$ of T and all the outdegrees in T. We check in $\mathcal{O}(n)$ time, whether there exists an index i, $1 \le i \le n-8k-1$, such that $d^+(v_{i+8k+1}) \le d^+(v_i) + 2k$. If such an index i is found, we infer that $\{v_i, v_{i+1}, \ldots, v_{i+8k+1}\}$ a $(8k+2, 2k)$-degree tangle, implying that $\mathbf{pw}(T) > 2k$ by Lemma 3.8 and, consequently, that $\mathbf{ctw}(T) > k$ by Lemma 2.13. Hence, in this case the algorithm can conclude that $\mathbf{ctw}(T) > k$. We proceed with the assumption that this did not take place, i.e., $d^+(v_{i+8k+1}) > d^+(v_i) + 2k$ for all $1 \le i \le n-8k-1$.

For an index i, $1 \le i \le n-8k$, let us define a multidigraph H_i as follows. Start with $T[\{v_i, v_{i+1}, \ldots, v_{i+8k}\}]$ and add two vertices: v_{prefix} and v_{suffix} that correspond to the prefix $\sigma[i-1]$ and suffix $V(T) \setminus \sigma[i+8k]$. For every $j \in \{i, i+1, \ldots, i+8k\}$, add $\min(k+1, |E(\sigma[i-1], \{v_j\})|)$ arcs from v_{prefix} to v_j, and $\min(k+1, |E(\{v_j\}, V(T) \setminus \sigma[i+8k])|)$ arcs from v_j to v_{suffix}. Finally, add $\min(k+1, |E(\sigma[i-1], V(T) \setminus \sigma[i+8k])|)$ arcs from v_{prefix} to v_{suffix}. Note that H_i defined in this manner has size polynomial in k.

The algorithm proceeds as follows. We iterate through consecutive indices i, maintaining the graph H_i. Observe that H_i can be maintained with $\mathcal{O}(n)$ update times, since each update requires inspection of incidence relation between v_i and the whole $V(T)$, and between v_{i+8k+1} and the whole $V(T)$. Thus, the total time spent on maintaining H_i is $\mathcal{O}(n^2)$. For each index i, we enumerate all the k-cuts (X', Y') of H_i such that $v_{\text{prefix}} \in X'$ and $v_{\text{suffix}} \in Y'$ using Lemma 5.8. This enumeration takes time proportional to their number times a polynomial of the size of H_i, that is, times a polynomial of k. For each enumerated k-cut (X', Y') we construct in $\mathcal{O}(n)$ time one k-cut (X, Y) of T equal to $(\sigma[i-1] \cup (X' \setminus v_{\text{prefix}}), (Y' \setminus v_{\text{suffix}}) \cup (V(T) \setminus \sigma[i+8k])$. By the definition of H_i we infer that (X, Y) is indeed a k-cut of T, and moreover $|E(X,Y)| = |E(X', Y')|$, so the size of the cut output by the algorithm of Lemma 5.8 can be stored as $|E(X,Y)|$. We store all the k-cuts constructed so far as binary vectors of length n in a prefix tree (trie). Thus in $\mathcal{O}(n)$ time we can check whether (X,Y) has not been already found, in which case it should be ignored, and otherwise we add it to the prefix tree in $\mathcal{O}(n)$ time. If the total number of constructed k-cuts exceed B at any point of the construction, we terminate the algorithm and provide the answer that $|\mathcal{N}(T,k)| > B$. Otherwise, we output all the constructed k-cuts. Since maintenance of graph H_i take $\mathcal{O}(n^2)$ time in total, and each next k-cut is identified and constructed within time $\mathcal{O}(k^{\mathcal{O}(1)} + n)$, for the claimed running time it suffices to show that each k-cut of T is found in this procedure at most $\mathcal{O}(k)$ times.

It remains to argue that (i) in case when the algorithm is providing the family of k-cuts, in fact every k-cut of T is contained in this family, (ii) each k-cut of T is constructed at most $\mathcal{O}(k)$ times. We prove both of the claims at the same time. Let then (X, Y) be a k-cut of T and without loss of generality assume that X and Y are nonempty, since k-cuts $(\emptyset, V(T))$ and $(V(T), \emptyset)$ are enumerated exactly once, for $i = 1$ and $i = n-8k$ respectively. Let α be the maximum index of a vertex of X in σ, and β be the minimum index of a vertex of Y in σ. We claim that $\beta - 1 \le \alpha \le \beta + 8k$. Observe that if this claim is proven, then both conditions (i) and (ii) follow: cut (X, Y) is constructed exactly when considering indices i such that $i \le \beta$ and $i + 8k \ge \alpha$, and the claimed inequalities show that the number of such indices is between 1 and $8k+2$.

We first show that $\beta - 1 \leq \alpha$. Consider two cases. If $\beta = 1$, then in fact all the elements of X have indices larger than β, so in particular $\alpha > \beta$. Otherwise $\beta > 1$, and by the minimality of β we have that $x_{\beta-1} \in X$. Consequently, $\alpha \geq \beta - 1$.

We are left with proving that $\alpha \leq \beta + 8k$. For the sake of contradiction assume that $\alpha > \beta + 8k$, so in particular $1 \leq \beta \leq n - 8k - 1$. By the assumption that $d^+(v_{i+8k+1}) > d^+(v_i) + 2k$ for all $1 \leq i \leq n - 8k - 1$, we have that $d^+(v_\alpha) \geq d^+(v_{\beta+8k+1}) > d^+(v_\beta) + 2k$. By Lemma 2.2 we infer that there exist $2k$ vertex-disjoint paths of length 2 from v_α to v_β. Since $v_\alpha \in X$ and $v_\beta \in Y$, each of these paths must contain an arc from $E(X, Y)$. This is a contradiction with the assumption that (X, Y) is a k-cut. □

In our dynamic programming algorithms we need to define what does it mean that one k-cut is a possible successor of another k-cut. This is encapsulated in the following definition.

Definition 5.10. *Let T be a digraph and (X_1, Y_1) and (X_2, Y_2) be two partitions of $V(G)$. We say that cut (X_2, Y_2)* extends cut (X_1, Y_1) using vertex v *if there is one vertex $v \in Y_1$ such that $X_2 = X_1 \cup \{v\}$ and, equivalently, $Y_2 = Y_1 \setminus \{v\}$.*

The following lemma shows that the relation of extension can be computed efficiently within the enumeration algorithm of Lemma 5.9.

Lemma 5.11. *If the algorithm of Lemma 5.9, run on a semi-complete digraph T for parameters k, B, provides the family $\mathcal{N}(T, k)$, then for each k-cut of T there are at most $8k + 1$ k-cuts of T that extend it. Moreover, the algorithm of Lemma 5.9 can within the same running time in addition construct for each k-cut of T a list of pointers to all the k-cuts that extend it, together with vertices used in these extensions.*

Proof. We only add one additional subroutine to the algorithm of Lemma 5.9 which computes the lists after the enumeration has been concluded. Assume that during enumeration we have constructed a k-cut (X, Y). Let α be the maximum index of a vertex of X in σ, and let β be the minimum index of a vertex of Y in σ (we take $\alpha = 0$ if $X = \emptyset$ and $\beta = n + 1$ if $Y = \emptyset$). In the proof of Lemma 5.9 we have proven that $\beta - 1 \leq \alpha \leq \beta + 8k$ (this claim is trivial for $X = \emptyset$ or $Y = \emptyset$), unless the algorithm already provided a negative answer. Let (X', Y') be a k-cut that extends (X, Y), and let v_γ be the vertex used in this extension; thus $\{v_\gamma\} = X' \cap Y$. Define indices α', β' in the same manner for (X', Y'). Note that they satisfy the same inequality, i.e. $\beta' - 1 \leq \alpha' \leq \beta' + 8k$, and moreover $\alpha \leq \alpha'$ and $\beta \leq \beta'$.

We now claim that $\beta \leq \gamma \leq \beta + 8k$. The first inequality follows from the fact that $v_\gamma \in Y$. For the second inequality, assume for the sake of contradiction that $\gamma > \beta + 8k$. Then $\beta \neq \gamma$, and since v_γ is the only vertex that belongs to Y but not to Y', we have that $v_\beta \in Y'$. We infer that $\beta' = \beta$. On the other hand, $v_\gamma \in X'$ implies that $\alpha' \geq \gamma$. Therefore

$$\beta' = \beta < \gamma - 8k \leq \alpha' - 8k,$$

which is a contradiction with the fact that $\alpha' \leq \beta' + 8k$.

Hence, there are only $8k + 1$ possible candidates for k-cuts that extend (X, Y), that is $(X \cup \{v_\gamma\}, Y \setminus \{v_\gamma\})$ for $\beta \leq \gamma \leq \beta + 8k$ and $v_\gamma \in Y$. For each of these candidates we may test in $\mathcal{O}(n)$ time whether it belongs to enumerated family $\mathcal{N}(T, k)$, since $\mathcal{N}(T, k)$ is stored in a prefix tree; note that computing index β also takes $\mathcal{O}(n)$ time. Hence, construction of the lists takes additional $\mathcal{O}(B \cdot k \cdot n)$ time. □

5.3.2 k-cuts of a transitive tournament and partition numbers

For a nonnegative integer n, a partition of n is a multiset of positive integers whose sum is equal to n. The partition number $p(n)$ is equal to the number of different partitions of n. Partition numbers were studied extensively in analytic combinatorics, and there are sharp estimates on their values. In particular, we will use the following:

Lemma 5.12 ([20, 36])**.** *There exists a constant A such that for every nonnegative k it holds that $p(k) \leq \frac{A}{k+1} \cdot \exp(C\sqrt{k})$, where $C = \pi\sqrt{\frac{2}{3}}$.*

We remark that the original proof of Hardy and Ramanujan [36] shows moreover that the optimal constant A tends to $\frac{1}{4\sqrt{3}}$ as k goes to infinity, i.e., $\lim_{k\to+\infty} \frac{p(k)\cdot(k+1)}{\exp(C\sqrt{k})} = \frac{1}{4\sqrt{3}}$. From now on, we adopt constants A, C given by Lemma 5.12 in the notation. We use Lemma 5.12 to obtain the following result, which is the core observation of this section.

Lemma 5.13. *Let T be a transitive tournament on n vertices and k be a nonnegative integer. Then T has at most $A \cdot \exp(C\sqrt{k}) \cdot (n+1)$ k-cuts, where A, C are defined as in Lemma 5.12.*

Proof. We prove that for any number a, $0 \leq a \leq n$, the number of k-cuts (X, Y) such that $|X| = a$ and $|Y| = n - a$, is bounded by $A \cdot \exp(C\sqrt{k})$; summing through all the possible values of a proves the claim.

We naturally identify the vertices of T with numbers $1, 2, \ldots, n$, such that arcs of T are directed from larger numbers to smaller, i.e., we order the vertices as in the reversed topological ordering of T. Let us fix some k-cut (X, Y) such that $|X| = a$ and $|Y| = n - a$. Let $x_1 < x_2 < \ldots < x_a$ be the vertices of X.

Let $m_i = x_{i+1} - x_i - 1$ for $i = 0, 1, \ldots, a$; we use convention that $x_0 = 0$ and $x_{a+1} = n + 1$. In other words, m_i is the number of elements of Y that are between two consecutive elements of X. Observe that every element of Y between x_i and x_{i+1} is the head of exactly $a - i$ arcs directed from X to Y: the tails are $x_{i+1}, x_{i+2}, \ldots, x_a$. Hence, the total number of arcs directed from X to Y is equal to $k' = \sum_{i=0}^{a} m_i \cdot (a - i) = \sum_{i=0}^{a} m_{a-i} \cdot i \leq k$.

We define a partition of k' as follows: we take m_{a-1} times number 1, m_{a-2} times number 2, and so on, up to m_0 times number a. Clearly, a k-cut of T defines a partition of k' in this manner. We now claim that knowing a and the partition of k', we can uniquely reconstruct the k-cut (X, Y) of T, or conclude that this is impossible. Indeed, from the partition we obtain all the numbers $m_0, m_1, \ldots, m_{a-1}$, while m_a can be computed as $(n - a) - \sum_{i=0}^{a-1} m_i$. Hence, we know exactly how large must be the intervals between consecutive elements of X, and how far is the first and the last element of X from the respective end of the ordering, which uniquely defines sets X and Y. The only possibilities of failure during reconstruction are that (i) the numbers in the partition are larger than a, or (ii) computed m_a turns out to be negative; in these cases, the partition does not correspond to any k-cut. Hence, we infer that the number of k-cuts of T having $|X| = a$ and $|Y| = n - a$ is bounded by the sum of partition numbers of nonnegative integers smaller or equal to k, which by Lemma 5.12 is bounded by $(k + 1) \cdot \frac{A}{k+1} \cdot \exp(C\sqrt{k}) = A \cdot \exp(C\sqrt{k})$. $\square$

5.3.3 k-cuts of semi-complete digraphs with a small FAS

We have the following simple fact.

Lemma 5.14. *Assume that T is a semi-complete digraph with a feedback arc set F of size at most k. Let T' be a transitive tournament on the same set of vertices, with vertices ordered as in any topological ordering of $T \setminus F$. Then every k-cut of T is also a $2k$-cut of T'.*

Proof. The claim follows directly from the observation that if (X, Y) is a k-cut in T, then at most k additional arcs directed from X to Y can appear after introducing arcs in T' in place of deleted arcs from F. $\square$

From Lemmata 5.13 and 5.14 we obtain the following corollary.

Corollary 5.15. *If T is a semi-complete digraph with n vertices that has a feedback arc set of size at most k, then the number of k-cuts of T is bounded by $A \cdot \exp(C\sqrt{2k}) \cdot (n+1)$.*

5.3.4 k-cuts of semi-complete digraphs of small cutwidth

To bound the number of k-cuts of semi-complete digraphs of small cutwidth, we need the following auxiliary combinatorial result.

Lemma 5.16. *Let (X, Y) be a partition of $\{1, 2, \ldots, n\}$ into two sets. We say that a pair (a, b) is* bad *if $a < b$, $a \in Y$ and $b \in X$. Assume that for every integer t there are at most k bad pairs (a, b) such that $a \leq t < b$. Then the total number of bad pairs is at most $k(1 + \ln k)$.*

Proof. Let $y_1 < y_2 < \ldots < y_p$ be the elements of Y. Let m_i be equal to the total number of elements of X that are greater than y_i. Note that m_i is exactly equal to the number of bad pairs whose first element is equal to y_i, hence the total number of bad pairs is equal to $\sum_{i=1}^{p} m_i$. Clearly, sequence (m_i) is non-increasing, so let p' be the last index for which $m_{p'} > 0$. We then have that the total number of bad pairs is equal to $\sum_{i=1}^{p'} m_i$. Moreover, observe that $p' \leq k$, as otherwise there would be more than k bad pairs (a, b) for which $a \leq y_{p'} < b$: for a we can take any y_i for $i \leq p'$ and for b we can take any element of X larger than $y_{p'}$.

We claim that $m_i \leq k/i$ for every $1 \leq i \leq p'$. Indeed, observe that there are exactly $i \cdot m_i$ bad pairs (a, b) for $a \leq y_i$ and $b > y_i$: a can be chosen among i distinct integers $y_1, y_2, \ldots, y_i$, while b can be chosen among m_i elements of X larger than y_i. By the assumption we infer that $i \cdot m_i \leq k$, so $m_i \leq k/i$. Concluding, we have that the total number of bad pairs is bounded by $\sum_{i=1}^{p'} m_i \leq \sum_{i=1}^{p'} k/i = k \cdot H(p') \leq k \cdot H(k) \leq k(1 + \ln k)$, where $H(k) = \sum_{i=1}^{k} 1/i$ is the harmonic function. $\square$

The following claim applies Lemma 5.16 to the setting of semi-complete digraphs.

Lemma 5.17. *Assume that T is a semi-complete digraph on n vertices that admits an ordering of vertices $(v_1, v_2, \ldots, v_n)$ of width at most k. Let T' be a transitive tournament on the same set of vertices, where $(v_i, v_j) \in E(T')$ if and only if $i > j$. Then every k-cut of T is a $2k(1 + \ln 2k)$-cut of T'.*

Proof. Without loss of generality we assume that T is in fact a tournament, as deleting any of two opposite arcs connecting two vertices can only make the set of k-cuts of T larger, and does not increase the width of the ordering.

Identify vertices $v_1, v_2, \ldots, v_n$ with numbers $1, 2, \ldots, n$. Let (X, Y) be a k-cut of T. Note that arcs of T' directed from X to Y correspond to bad pairs in the sense of Lemma 5.16: every arc $(b, a) \in E(T')$ such that $a < b$, $a \in Y$, and $b \in X$, corresponds to a bad pair (a, b), and vice versa. Therefore, by Lemma 5.16 it suffices to prove that for every integer t, the number of arcs $(b, a) \in E(T')$ such that $a \leq t < b$, $a \in Y$, and $b \in X$, is bounded by $2k$. We know that the number of such arcs in T is at most k, as there are at most k arcs directed from X to Y in T in total. Moreover, as the considered ordering of T has cutwidth at most k, at most k arcs between vertices from $\{1, 2, \ldots, t\}$ and $\{t+1, \ldots, n\}$ can be directed in different directions in T and in T'. We infer that the number of arcs $(b, a) \in E(T')$ such that $a \leq t < b$, $a \in Y$, and $b \in X$, is bounded by $2k$, and so the lemma follows. $\square$

From Lemmata 5.13 and 5.17 we obtain the following corollary.

Corollary 5.18. *Every semi-complete digraph on n vertices and of cutwidth at most k, has at most $A \cdot \exp(2C\sqrt{k(1+\ln 2k)}) \cdot (n+1)$ k-cuts.*

5.3.5 k-cuts of semi-complete digraphs with an ordering of small cost

We firstly show the following lemma that proves that semi-complete digraphs with an ordering of small cost have even smaller cutwidth.

Lemma 5.19. *Let T be a semi-complete digraph on n vertices that admits an ordering $(v_1, v_2, \ldots, v_n)$ of cost at most k. Then the width of this ordering is at most $(4k)^{2/3}$.*

Proof. We claim that for every integer $t \geq 0$, the number of arcs in T directed from the set $\{v_1, \ldots, v_t\}$ to $\{v_{t+1}, \ldots, v_n\}$ is at most $(4k)^{2/3}$. Let ℓ be the number of such arcs; without loss of generality assume that $\ell > 0$. Observe that at most one of these arcs may have length 1, at most 2 may have length 2, etc., up to at most $\lfloor\sqrt{\ell}\rfloor - 1$ may have length $\lfloor\sqrt{\ell}\rfloor - 1$. It follows that at most $\sum_{i=1}^{\lfloor\sqrt{\ell}\rfloor-1} i \leq \ell/2$ of these arcs may have length smaller than $\lfloor\sqrt{\ell}\rfloor$. Hence, at least $\ell/2$ of the considered arcs have length at least $\lfloor\sqrt{\ell}\rfloor$, so the total sum of lengths of arcs is at least $\frac{\ell\cdot\lfloor\sqrt{\ell}\rfloor}{2} \geq \frac{\ell^{3/2}}{4}$. We infer that $k \geq \frac{\ell^{3/2}}{4}$, which means that $\ell \leq (4k)^{2/3}$. □

Lemma 5.19 ensures that only $(4k)^{2/3}$-cuts are interesting from the point of view of dynamic programming. Moreover, from Lemma 5.19 and Corollary 5.18 we can derive the following statement that bounds the number of states of the dynamic program.

Corollary 5.20. *If T is a semi-complete digraph with n vertices that admits an ordering of cost at most k, then the number of $(4k)^{2/3}$-cuts of T is bounded by $A \cdot \exp(2C \cdot (4k)^{1/3} \cdot \sqrt{1+\ln(2\cdot(4k)^{2/3})}) \cdot (n+1)$.*

5.4 The algorithms

We firstly show how using the approach one can find a simple algorithm for Feedback Arc Set.

Theorem 5.21. *There exists an algorithm that, given a semi-complete digraph T on n vertices and an integer k, in time $\mathcal{O}(\exp(C\sqrt{2k}) \cdot k^{\mathcal{O}(1)} \cdot n^2)$ either finds a feedback arc set of T of size at most k or correctly concludes that this is impossible, where $C = \pi\sqrt{\frac{2}{3}}$.*

Proof. We apply the algorithm of Lemma 5.9 in T for parameter k and the bound $A\cdot\exp(C\sqrt{2k})\cdot(n+1)$ given by Corollary 5.15. If the algorithm concluded that $\mathbf{ctw}(T) > k$, then also the minimum feedback arc set must be of size more than k, and we may provide a negative answer. Similarly, if the algorithm concluded that $|\mathcal{N}(T,k)| > A\cdot\exp(C\sqrt{2k})\cdot(n+1)$, by Corollary 5.15 we may also provide a negative answer. Hence, from now on we assume that we have the set $\mathcal{N} := \mathcal{N}(T,k)$ and we know that $|\mathcal{N}| \leq A \cdot \exp(C\sqrt{2k}) \cdot (n+1)$. Note that application of Lemma 5.9 takes $\mathcal{O}(\exp(C\sqrt{2k}) \cdot k^{\mathcal{O}(1)} \cdot n^2)$ time.

We now describe a dynamic programming procedure that computes the size of optimal feedback arc set basing on the set of k-cuts $\mathcal{N}$. We define an auxiliary weighted digraph D with vertex set $\mathcal{N}$. Intuitively, a vertex from $\mathcal{N}$ corresponds to a partition into prefix and suffix of the ordering.

We define arcs of D as follows. For every pair of k-cuts (X_1, Y_1), (X_2, Y_2) such that (X_2, Y_2) extends (X_1, Y_1) using vertex v, we put an arc from cut (X_1, Y_1) to cut (X_2, Y_2), where the weight of this arc is equal to $|E(X_1, \{v\})|$, that is, the number of arcs that cease to be be directed from the left side to the right side of the partition when moving v between these parts. Construction of arcs D may be performed in $\mathcal{O}(|\mathcal{N}| \cdot k^{\mathcal{O}(1)} \cdot n)$ time, assuming that the enumeration of Lemma 5.9 constructed also extension lists using Lemma 5.11. Moreover, the weight of each arc can be computed in $\mathcal{O}(n)$ time by examining outneighbors of v, hence the total time spent on computing weights of arcs is $\mathcal{O}(|\mathcal{N}| \cdot k \cdot n)$. Summarizing, the whole digraph D may be constructed in $\mathcal{O}(|\mathcal{N}| \cdot k^{\mathcal{O}(1)} \cdot n) = \mathcal{O}(\exp(C\sqrt{2k}) \cdot k^{\mathcal{O}(1)} \cdot n^2)$ time, and has $|\mathcal{N}|$ vertices and $\mathcal{O}(|\mathcal{N}| \cdot k)$ arcs.

Observe that a path from vertex $(\emptyset, V(T))$ to a vertex $(V(T), \emptyset)$ of total weight ℓ defines an ordering of vertices of T that has exactly ℓ forward arcs — each of these arcs was taken into account while moving its head from the right side of the partition to the left side. On the other hand, every ordering of vertices of T that has exactly $\ell \leq k$ forward arcs defines a path from $(\emptyset, V(T))$ to $(V(T), \emptyset)$ in D of total weight ℓ; note that all partitions into prefix and suffix in this ordering are k-cuts, so they constitute legal vertices in D. Hence, we need to check whether vertex $(V(T), \emptyset)$ can be reached from $(\emptyset, V(T))$ by a path of total weight at most k. This, however, can be done in time $\mathcal{O}((|V(D)| + |E(D)|) \log |V(D)|) = \mathcal{O}(\exp(C\sqrt{2k}) \cdot k^{\mathcal{O}(1)} \cdot n \log n)$ using Dijkstra's algorithm. The feedback arc set of size at most k can be easily retrieved from the constructed path in $\mathcal{O}(n^2)$ time. □

We remark that it is straightforward to adapt the algorithm of Theorem 5.21 to the weighted case, where all the arcs are assigned a real weight larger or equal to 1 and we parametrize by the target total weight of the solution. As the minimum weight is at least 1, we may still consider only k-cuts of the digraph where the weights are forgotten. On this set we employ a modified dynamic programming routine, where the weights of arcs in digraph D are not simply the number of arcs in $E(\{v\}, X_1)$, but their total weight.

We now proceed to the main result of this section, i.e., the subexponential algorithm for cutwidth of a semi-complete digraph.

Theorem 5.22. *There exists an algorithm that, given a semi-complete digraph T on n vertices and an integer k, in time $2^{\mathcal{O}(\sqrt{k \log k})} \cdot n^2$ either computes a vertex ordering of width at most k or correctly concludes that this is impossible.*

Proof. We apply the algorithm of Lemma 5.9 in T for parameter k and the bound

$$A \cdot \exp(2C\sqrt{k(1 + \ln 2k)}) \cdot (n + 1) = 2^{\mathcal{O}(\sqrt{k \log k})} \cdot n$$

given by Corollary 5.18. If the algorithm concluded that $\mathbf{ctw}(T) > k$, then we may provide a negative answer. Similarly, if the algorithm concluded that

$$|\mathcal{N}(T, k)| > A \cdot \exp(2C\sqrt{k(1 + \ln 2k)}) \cdot (n + 1),$$

by Corollary 5.18 we may also providing a negative answer. Hence, from now on we assume that we have the set $\mathcal{N} := \mathcal{N}(T, k)$ and we know that $|\mathcal{N}| \leq 2^{\mathcal{O}(\sqrt{k \log k})} \cdot n$. Note that application of Lemma 5.9 takes $\mathcal{O}(2^{\mathcal{O}(\sqrt{k \log k})} \cdot n^2)$ time.

We now describe a dynamic programming routine that basing on the set $\mathcal{N}$ computes an ordering of width at most k, or correctly concludes that it is impossible. The routine is very similar to that of Theorem 5.21, so we describe only the necessary modifications.

We define an auxiliary digraph D on the vertex set $\mathcal{N}$ exactly in the same manner as in the proof of Theorem 5.21, but this time D is unweighted. Similarly as before, D may be constructed in time $\mathcal{O}(|\mathcal{N}| \cdot k^{\mathcal{O}(1)} \cdot n) = 2^{\mathcal{O}(\sqrt{k \log k})} \cdot n^2$, and has $|\mathcal{N}|$ vertices and $\mathcal{O}(|\mathcal{N}| \cdot k)$ arcs. Clearly, paths in D from $(\emptyset, V(T))$ to $(V(T), \emptyset)$ correspond to orderings of $V(T)$ of cutwidth at most k. Therefore, it suffices check whether in D there exists a path from $(\emptyset, V(T))$ to $(V(T), \emptyset)$ using depth-first search, which takes $\mathcal{O}(|V(D)| + |E(D)|) = 2^{\mathcal{O}(\sqrt{k \log k})} n$ time. □

Finally, we present how the framework can be applied to the OLA problem.

Theorem 5.23. *There exists an algorithm that, given a semi-complete digraph T on n vertices and an integer k, in time $2^{\mathcal{O}(k^{1/3}\sqrt{\log k})} \cdot n^2$ either computes a vertex ordering of cost at most k, or correctly concludes that it is not possible.*

Proof. We apply the algorithm of Lemma 5.9 in T for parameter $(4k)^{2/3}$ and the bound

$$A \cdot \exp(2C \cdot (4k)^{1/3} \cdot \sqrt{1 + \ln(2 \cdot (4k)^{2/3})}) \cdot (n+1) = 2^{\mathcal{O}(k^{1/3}\sqrt{\log k})} \cdot n$$

given by Corollary 5.20. If the algorithm concluded that $\mathbf{ctw}(T) > (4k)^{2/3}$, then by Lemma 5.19 we may provide a negative answer. Similarly, if the algorithm concluded that

$$|\mathcal{N}(T, (4k)^{2/3})| > A \cdot \exp(2C \cdot (4k)^{1/3} \cdot \sqrt{1 + \ln(2 \cdot (4k)^{2/3})}) \cdot (n+1),$$

by Corollary 5.20 we may also provide a negative answer. Hence, from now on we assume that we have the set $\mathcal{N} := \mathcal{N}(T, (4k)^{2/3})$ and we know that $|\mathcal{N}| \leq 2^{\mathcal{O}(k^{1/3}\sqrt{\log k})} \cdot n$. Note that application of Lemma 5.9 takes $2^{\mathcal{O}(k^{1/3}\sqrt{\log k})} \cdot n^2$ time.

In order to construct the dynamic programming routine, we proceed very similarly to the proof of Theorem 5.21. Define the same auxiliary digraph D on the vertex set $\mathcal{N}$, where we put an arc from (X_1, Y_1) to (X_2, Y_2) if and only if (X_2, Y_2) extends (X_1, Y_1); this time, the weight of this arc is equal to $|E(X_1, Y_1)|$. As in the proof of Theorem 5.21, the digraph D may be constructed in time $\mathcal{O}(|\mathcal{N}| \cdot k^{\mathcal{O}(1)} \cdot n)$, and has $|\mathcal{N}|$ vertices and $\mathcal{O}(|\mathcal{N}| \cdot k)$ arcs; note here that we do not need to additionally compute the weights of arcs in D, since values $|E(X, Y)|$ have been provided together with the cuts (X, Y) by the enumeration algorithm.

Now observe that paths from $(\emptyset, V(T))$ to $(V(T), \emptyset)$ of total weight $\ell \leq k$ correspond one-to-one to orderings with cost ℓ: the weight accumulated along the path computes correctly the cost of the ordering due to Lemma 5.6. Note that Lemma 5.19 ensures that in an ordering of cost at most k, the only feasible partitions into prefix and suffix of the ordering are in $\mathcal{N}$, so they constitute legal vertices in D. Hence, we may apply Dijkstra's algorithm to check whether vertex $(V(T), \emptyset)$ is reachable from $(\emptyset, V(T))$ via a path of total weight at most k, and this application takes $\mathcal{O}((|V(D)| + |E(D)|) \log |V(D)|) = 2^{\mathcal{O}(k^{1/3}\sqrt{\log k})} \cdot n \log n$ time. The corresponding ordering may be retrieved from this path in $\mathcal{O}(n)$ time. □

Similarly to Theorem 5.21, it is also straightforward to adapt the algorithm of Theorem 5.23 to the natural weighted variant of the problem, where each arc is assigned a real weight larger or equal to 1, each arc directed forward in the ordering contributes to the cost with its weight multiplied by the length of the arc, and we parametrize by the total target cost. One needs also to maintain the total weight of the cut in the enumeration algorithm of Lemma 5.9 to avoid its recomputation for every arc of D, which done by brute-force would increase the running time from quadratic in terms of n to cubic.

6 Algorithms for containment testing

In this section we utilize the results of the previous sections to give fast algorithms for containment testing. We would like to design a dynamic programming routine that on decomposition of width p of a semi-complete digraph T on n vertices, works in time $f(|H|, p) \cdot n^2$ for some (possibly small) function f. Existence of such an algorithm for the topological subgraph and minor relations follows immediately from Theorem 2.16, as the definition of containing H as a topological subgraph or a minor can be trivially expressed by a $\mathbf{MSO}_1$ formula of length depending on $|H|$ only. For the immersion relation one can use the following result of Ganian et al. [34].

Theorem 6.1 ([34])**.** *For every ℓ there exists an* $\mathbf{MSO}_1$ *formula $\pi_\ell(s_1, s_2, \ldots, s_\ell, t_1, t_2, \ldots, t_\ell)$ that for a digraph with distinguished vertices $s_1, s_2, \ldots, s_\ell, t_1, t_2, \ldots, t_\ell$ (some of which are possibly equal) asserts whether there exists a family of arc-disjoint paths $P_1, P_2, \ldots, P_\ell$ such that P_i begins in s_i and ends in t_i, for $i = 1, 2, \ldots, \ell$.*

Therefore, in order to construct a formula expressing that a digraph T contains a digraph H as an immersion, we can quantify existentially the images of vertices of H, verify that they are pairwise different, and then use formula $\pi_{|E(H)|}$ to check existence of arc-disjoint paths between corresponding images. Unfortunately, for Theorem 2.16 only very crude upper bounds on the obtained function f can be given. Essentially, the engine behind Theorem 2.16 is a translation of formulas of **MSO** logic on trees to tree automata. It is known that the dependence of the size of the automaton obtained in the translation on the size of the formula is roughly q-times exponential, where q is the quantifier rank of the formula. Moreover, this is unavoidable in the following sense: the size of the automaton cannot be bounded by any elementary function of the length of the formula. Since formulas expressing the topological subgraph and minor relations have constant quantification rank, the function $f(\cdot, \cdot)$ given by an application of Theorem 2.16 will be in fact elementary. As far as the immersion relation is concerned, the quantification rank of the formula given by Theorem 6.1 is linear in ℓ, and thus we cannot even claim elementary running time bounds.

Instead, in order to give explicit bounds on the running time of topological subgraph, immersion, and minor testing, we design explicit dynamic programming routines. The routines are presented in Section 10, and here we only state the results. We remark that the routines can in fact check more involved properties, which will be useful in applications.

Theorem 6.2. *There exists an algorithm that given a digraph H with k vertices, ℓ arcs, a set $F \subseteq E(H)$ of constraints, and a semi-complete digraph T on n vertices together with its path decomposition of width p, checks whether H is topologically contained in T with constraints F in time $2^{\mathcal{O}((p+k+\ell)\log p)} \cdot n$.*

Theorem 6.3. *There exists an algorithm that, given a rooted digraph $(H; v_1, v_2, \ldots, v_t)$ with k vertices and ℓ arcs, and a semi-complete rooted digraph $(T; w_1, w_2, \ldots, w_t)$ on n vertices together with its path decomposition of width p, checks whether H can be immersed into T while preserving roots in time $2^{\mathcal{O}(k\log p + p\ell(\log \ell + \log p))} \cdot n$.*

It may be somewhat surprising that the algorithms of Theorems 6.2 and 6.3 work in time linear in n, while representation of T using adjacency matrix uses $\mathcal{O}(n^2)$ space. Note however, that a representation of a path decomposition of width p uses only $\mathcal{O}(pn)$ space. In these theorems we assume that the algorithm is given access to an adjacency matrix representing T that can be queried in constant time. Both algorithms perform one linear scan through the given path decomposition using the adjacency matrix as a black-box, and performing only a linear number of queries on it.

By pipelining Lemma 2.4 and Theorem 6.2 we obtain also the routine for the minor relation.

Theorem 6.4. *There exists an algorithm that given digraph H with k vertices and ℓ arcs, and a semi-complete digraph T together with its path decomposition of width p, checks in time $2^{\mathcal{O}((p+k+\ell)(\log p + \log k + \log \ell))} \cdot n$, whether H is a minor of T.*

We are ready to provide formal descriptions of the containment testing algorithms.

Theorem 6.5. *There exists an algorithm that, given a semi-complete T on n vertices and a digraph H with $k = |H|$, in time $2^{\mathcal{O}(k \log k)} \cdot n^2$ checks whether H is topologically contained in T.*

Proof. We run the algorithm given by Theorem 4.12 for parameters $20k$ and $520k$, which either returns a $(520k + 2, 20k)$-degree tangle, a $(20k + 1, 20k)$-matching tangle or a decomposition of width at most $560k$. If the last is true, we run the dynamic programming routine of Theorem 6.2, which works in $2^{\mathcal{O}(k \log k)} \cdot n$ time. However, if the approximation algorithm returned an obstacle, by Lemmas 3.9, 3.12 and 3.3 we can provide a positive answer: existence of a $(520k + 2, 20k)$-degree tangle or a $(20k + 1, 20k)$-matching tangle ensures that H is topologically contained in T. □

By plugging in the dynamic programming routine for immersion (Theorem 10.8 with no roots specified) instead of topological containment, we obtain the following:

Theorem 6.6. *There exists an algorithm that, given a semi-complete T on n vertices and a digraph H with $k = |H|$, in time $2^{\mathcal{O}(k^2 \log k)} \cdot n^2$ checks whether H can be immersed into T.*

Finally, in the same manner we can use the algorithm of Theorem 6.4 and substitute the usage of Lemma 3.3 with Lemma 3.4 to obtain the algorithm for testing the minor relation. Note that application of Lemma 3.4 requires providing a slightly larger jungle; hence, in the application of Theorem 4.12 we use parameters $45k$ and $1170k$ instead of $20k$ and $520k$.

Theorem 6.7. *There exists an algorithm that, given a semi-complete T on n vertices and a digraph H with $k = |H|$, in time $2^{\mathcal{O}(k \log k)} \cdot n^2$ checks whether H is a minor of T.*

7 Weak duality of width measures and containment relations

A direct consequence of the obtained combinatorial results is that we can relate the pathwidth of a semi-complete digraph with the size of the smallest excluded topological subgraph/minor, and its cutwidth with the size of the smallest excluded immersion. We remark that the pure existence of these relations was already discovered in the works of Fradkin and Seymour [31], and of Chudnovsky, Fradkin, and Seymour [11]; however, the quantitative relation is multiple exponential and unspecified. Our results imply explicit and asymptotically tight bounds: excluding H as a topological subgraph or a minor implies an $\mathcal{O}(|H|)$ upper bound on the pathwidth, and excluding H as an immersion implies an $\mathcal{O}(|H|^2)$ upper bound on cutwidth. The proofs follow closely the strategy presented in the proof of Theorem 6.5.

Theorem 7.1. *Let T be a semi-complete digraph that does not contain a digraph H as a topological subgraph. Then $\mathbf{pw}(T) \leq 560k$, where $k = |H|$.*

Proof. From Theorem 4.12 applied to T for parameters $20k$ and $520k$ we infer that either $\mathbf{pw}(T) \leq 560k$, or T contains a $(520k + 2, 20k)$-degree tangle or a $(20k + 1, 20k)$-matching tangle. Similarly as in the proof of Theorem 6.5, it follows from Lemmas 3.9, 3.12 and 3.3 that the existence of any of these two obstacles would imply that H is topologically contained in T, contradicting the assumption. Hence $\mathbf{pw}(T) \leq 560k$. □

Theorem 7.2. *Let T be a semi-complete digraph that does not contain a digraph H as a minor. Then $\mathbf{pw}(T) \leq 1260k$, where $k = |H|$.*

Proof. As in Theorem 6.7, we follow the same approach as in the proof of Theorem 7.1, but we use Theorem 4.12 with parameters $45k$ and $1170k$ and substitute the usage of Lemma 3.3 with Lemma 3.4. □

Theorem 7.3. *Let T be a semi-complete digraph that does not contain a digraph H as an immersion. Then $\mathbf{ctw}(T) \leq m(55k)$, where $k = |H|$ and $m(t) = 64t^2 + 18t + 1$.*

Proof. From Theorem 5.2 applied to T for parameter $55k$ we infer that either $\mathbf{ctw}(T) \leq m(55k)$, or T contains a $(m(55k) + 1)$-backwards tangle. Observe that $m(55k) + 1 > 64 \cdot 55^2 \cdot k^2 > 436^2 \cdot k^2 = 109^2(4k)^2$. Thus, by Lemma 3.3 the existence of a $(m(55k) + 1)$-backwards tangle would imply the existence of a $(4k, 4)$-short immersion jungle. By Lemma 3.3 this would imply that H can be immersed into T, contradicting the assumption. Hence $\mathbf{ctw}(T) \leq m(55k)$. $\square$

We conclude this section by observing that the upper bounds of Theorems 7.1, 7.2, and 7.3 are asymptotically tight in the following sense: the upper bounds on $\mathbf{pw}(T)$ in Theorems 7.1 and 7.2 have to be at least linear in k, and the upper bound on $\mathbf{ctw}(T)$ of Theorem 7.3 has to be at least quadratic in k. To see this, let S_{2k} be a *complete* digraph on $2k$ vertices, i.e., one where for every pair of distinct $u, v \in V(S_{2k})$ both arcs (u, v) and (v, u) are present in S_{2k}. Then $\mathbf{pw}(S_{2k}) = 2k - 1$ and $\mathbf{ctw}(S_{2k}) = k^2$. On the other hand, if H is any digraph on $2k + 1$ vertices that has $\mathcal{O}(k)$ arcs (for instance, it is a directed path on $2k + 1$ vertices), then $|H| = \mathcal{O}(k)$ and S_{2k} contains H neither as a topological subgraph, nor as a minor, nor as an immersion. By considering H and $T = S_{2k}$ in the statements of Theorems 7.1, 7.2, and 7.3 we infer the claimed tightness of our upper bounds.

8 Containment testing and meta-theorems

We observe that FPT algorithms for testing containment relations can be used also to prove meta-theorems of more general nature using the WQO results of Chudnovsky and Seymour [13] and of Kim and Seymour [42]. We explain this on the following example. Let Π be a class of digraphs and denote by $\Pi + kv$ the class of digraphs, from which one can delete at most k vertices to obtain a member of Π. We study the following problem:

$\Pi + kv$ Recognition

Input:	Digraph D and a non-negative integer k
Parameter:	k
Question:	Is there $S \subseteq V(D)$, $\lvert S\rvert \leq k$, such that $D \setminus S \in \Pi$?

We are interested in classes Π which are closed under immersion. For example the class of acyclic digraphs, or digraphs having cutwidth at most c, where c is some constant, are of this type (see Lemma 2.8). In particular, the parameterized Feedback Vertex Set in directed graphs is equivalent to $\Pi + kv$ Recognition for Π being the class of acyclic digraphs. Chudnovsky and Seymour [13] showed that immersion order on semi-complete digraphs is a well-quasi-order, see Theorem 2.5. Based on this result and the approximation algorithm for pathwidth, we are able to prove the following meta-theorem. Note that it seems difficult to obtain the results of this flavor using cutwidth, as cutwidth can decrease dramatically even when one vertex is deleted from the digraph.

Theorem 8.1. *Let Π be an immersion-closed class of semi-complete digraphs. Then $\Pi + kv$ Recognition is FPT on semi-complete digraphs.*

Proof. As Π is immersion-closed, by Theorem 2.5 we infer that Π can be characterized by admitting no member of a family of semi-complete digraphs $\{H_1, H_2, \ldots, H_r\}$ as an immersion, where $r = r(\Pi)$ depends only on the class Π. We will again make use of Theorem 6.1. For every $i \in \{1, 2, \ldots, r\}$ we construct an $\mathbf{MSO}_1$ formula $\varphi_i(X)$ with one free monadic vertex variable X that is true if and only if digraph $G \setminus X$ contains H_i as an immersion. We simply quantify existentially over the images of vertices of H_i, use the appropriate formula $\pi_{|E(H)|}$ for quantified variables to express existence of arc-disjoint paths, and at the end relativize the whole formula to the subdigraph induced by $V(T) \setminus X$. Hence, if we denote by $\psi_k(X)$ the assertion that $|X| \leq k$ (easily expressible in **FO** by a formula, whose length depends on k), the formula $\varphi = \exists_X \psi_k(X) \wedge \bigwedge_{i=1}^{r} \neg\varphi_i(X)$ is true exactly in semi-complete digraphs from which one can delete at most k vertices in order to obtain a semi-complete digraphs belonging to Π.

Observe that every member of class Π has pathwidth bounded by a constant depending on Π only, as by Theorem 4.12 and Lemmas 3.9, 3.12 and 3.3, a semi-complete digraph of large enough pathwidth contains a sufficiently large short jungle, in which one of the digraphs H_i is topologically contained, so also immersed. It follows that if the pathwidth of every member of Π is bounded by c_Π, then the pathwidth of every member of $\Pi + kv$ is bounded by $c_\Pi + k$. Therefore, we can apply the following WIN/WIN approach. We apply Theorem 4.12 for parameters k' and $4k'$ where $k' = c_\Pi + k$. This application takes time $g(k)|V(T)|^2$ and provides either an obstacle for admitting a path decomposition of width at most $c_\Pi + k$, which is sufficient to provide a negative answer, or a path decomposition of width at most $6(c_\Pi + k)$, on which we can run the algorithm given by Theorem 2.16 applied to formula φ. □

9 The algorithm for Rooted Immersion

In this section we apply the developed tools to solve the Rooted Immersion problem in semi-complete digraphs, i.e., testing whether one digraph H is an immersion of another semi-complete digraph T where some of the vertices have already prescribed images.

The algorithm of Theorem 6.6 cannot be used to solve Rooted Immersion because in the case when an obstacle is found, we are unable to immediately provide the answer: even though H can be found in the obstacle as an immersion, this embedding can have nothing to do with the roots. Therefore, we need to exploit the identified obstacle in a different way. Following the classical approach in such a case, we design an *irrelevant vertex rule*. That is, given the obstacle we find in polynomial time a vertex that can be assumed to be not used by some solution, and which therefore can be safely deleted from the graph. After this deletion we simply restart the algorithm. Thus at each step of this process the algorithm either solves the problem completely using dynamic programming on a path decomposition of small width, or removes one vertex of the graph; as a result, we run the approximation algorithm of Theorem 4.12 and identification of an irrelevant vertex at most n times.

The design of the irrelevant vertex rule needs very careful argumentation, because one has to argue that some vertex is omitted by some solution without having any idea about the shape of this solution, or even of existence of any solutions at all. Therefore, instead of short jungles that were the main tool used in Section 6, we use the original obstacle introduced by Fradkin and Seymour [31], namely the triple.

9.1 Irrelevant vertex in a triple

In this section we show how to identify a vertex that is irrelevant for the Rooted Immersion problem in a semi-complete graph with a sufficiently large triple. Let $p(k) = 80k^2 + 80k + 5$. We prove the following theorem.

Theorem 9.1. *Let $\mathcal{I} = ((H; u_1, u_2, \ldots, u_t), (T; v_1, v_2, \ldots, v_t))$ be an instance of the* Rooted Immersion *problem, where $k = |H|$ and T is a semi-complete graph on n vertices containing a $p(k)$-triple (A, B, C) disjoint with $\{v_1, v_2, \ldots, v_t\}$. Then it is possible in time $\mathcal{O}(p(k)^2 \cdot n^2)$ to* *identify a vertex $x \in B$ such that $\mathcal{I}$ is a YES instance of* Rooted Immersion *if and only if $\mathcal{I}' = ((H, u_1, u_2, \ldots, u_t), (T \setminus \{x\}, v_1, v_2, \ldots, v_t))$ is a YES instance.*

Before we proceed with the proof of Theorem 9.1, we need to make several auxiliary observations.

Let η be a solution to the Rooted Immersion instance and let $\mathcal{Q}$ be the family of paths being images of all the arcs in H, i.e., $\mathcal{Q} = \eta(E(H))$. We call an arc (a vertex) *used by a path* P if it is traversed by P. We say that an arc (a vertex) is *used by* $\mathcal{Q}$ if it is used by any path of $\mathcal{Q}$. We omit the family $\mathcal{Q}$ whenever it is clear from the context. An arc (a vertex) which is not used is called a *free* arc (vertex).

Observation 9.2. *If $\mathcal{Q}$ is a family of paths containing simple paths only, then every vertex in T is adjacent to at most k used incoming arcs and at most k used outgoing arcs.*

Proof. Otherwise, there is a path in the solution that visits that vertex at least two times. Therefore, there is a cycle on this path, which contradicts its simplicity. □

Let η be a solution to Rooted Immersion instance $\mathcal{I}$ that minimizes the total sum of length of paths in $\eta(E(H))$, and let $\mathcal{Q} = \eta(E(H))$. Firstly, we observe some easy properties of $\mathcal{Q}$.

Observation 9.3. *Every path from $\mathcal{Q}$ uses at most 2 arcs from the matching between C and A.*

Proof. Assume otherwise, that there is a path $P \in \mathcal{Q}$ that uses three arcs of the matching: (c_1, a_1), (c_2, a_2), (c_3, a_3), appearing in this order on the path. By Observation 9.2, for at most k vertices of $v \in B$ the arc (a_1, v) is used. For the same reason, for at most k vertices $v \in B$ the arc (v, c_3) is used. As $|B| > 2k$, there exists $v \in B$ such that (a_1, v) and (v, c_3) are not used. Now replace the part of P appearing between a_1 and c_3 with $a_1 \to v \to c_3$. We obtain a solution with smaller sum of lengths of the paths, a contradiction. □

Observation 9.4. *Every path from $\mathcal{Q}$ uses at most $2k + 4$ vertices from A.*

Proof. Assume otherwise, that there is a path $P \in \mathcal{Q}$ passing through at least $2k + 5$ vertices from A. By Observation 9.3, at most $2k$ of them are endpoints of used arcs of the matching between C and A. Therefore, there are at least 5 visited vertices, which are endpoints of an unused arc of the matching. Let us denote any 5 of them by a_1, a_2, a_3, a_4, a_5 and assume that they appear on P in this order. Let (c_5, a_5) be the arc of the matching between C and A that is incident to a_5. By the same reasoning as in the proof of Observation 9.3, there exists a vertex $v \in B$, such that (a_1, v) and (v, c_5) are unused arcs. Substitute the part of the path P between a_1 and a_5 by the path $a_1 \to v \to c_5 \to a_5$, which consists only of unused arcs. We obtain a solution with smaller sum of lengths of the paths, a contradiction. □

A symmetrical reasoning yields the following observation.

Observation 9.5. *Every path from $\mathcal{Q}$ uses at most $2k + 4$ vertices from C.*

Finally, we prove a similar property for B.

Observation 9.6. *Every path from $\mathcal{Q}$ uses at most 4 vertices from B.*

Proof. Assume otherwise, that there is a path $P \in \mathcal{Q}$ such that it passes through at least 5 vertices from B. Let us denote any 5 of them by b_1, b_2, b_3, b_4, b_5 and assume that they appear on P in this order. By Observation 9.2 there are at most k outgoing arcs incident to b_1 used, and there are at most k incoming arcs incident to b_5 used. Moreover, by Observation 9.3 there are at most $2k$ arcs of the matching between C and A used. As $p(k) > 4k$, we conclude that there is an unused arc of the matching (c, a), such that arcs (b_1, c) and (a, b_5) are also unused. Substitute the part of the path P between b_1 and b_5 by the path $b_1 \to c \to a \to b_5$. We obtain a solution with smaller sum of lengths of the paths, a contradiction. □

From Observations 9.4-9.6 we obtain the following corollary.

Corollary 9.7. *In B there are at most $4k$ vertices used by $\mathcal{Q}$, so in particular there are at least $5k+1$ vertices free from $\mathcal{Q}$. Moreover, within the matching between C and A there are at least $4k$ arcs having both endpoints free from $\mathcal{Q}$.*

We note that the Corollary 9.7 holds also for much larger values than $5k+1$, $4k$, respectively; we choose to state it in this way to show how many free vertices from B and free arcs of the matching we actually use in the proof of Theorem 9.1. We need one more auxiliary lemma that will prove to be useful.

Lemma 9.8. *Let $T = (V_1 \cup V_2, E)$ be a semi-complete bipartite digraph, i.e., a directed graph, where arcs are only between V_1 and V_2, but for every $v_1 \in V_1$ and $v_2 \in V_2$ at least one of the arcs (v_1, v_2) and (v_2, v_1) is present. Then at least one of the assertions holds:*

(a) for every $v_1 \in V_1$ there exists $v_2 \in V_2$ such that $(v_1, v_2) \in E$;

(b) for every $v_2 \in V_2$ there exists $v_1 \in V_1$ such that $(v_2, v_1) \in E$.

Proof. Assume that (a) does not hold. This means that there is some $v_0 \in V_1$ such that for all $v_2 \in V_2$ we have $(v_2, v_0) \in E$. Then we can always pick v_0 as v_1 in the statement of (b), so (b) holds. □

Observe that by reversing all the arcs we can obtain a symmetrical lemma, where we assert existence of inneighbors instead of outneighbors.

We are now ready to prove Theorem 9.1. Whenever we will refer to the *matching*, we mean the matching between C and A.

Proof of Theorem 9.1. To prove the theorem we give an algorithm that outputs a vertex $x \in B$, such that if there exists a solution to the the given instance, then there exists also a solution in which no path passes through x. The algorithm will run in time $\mathcal{O}(p(k)^2 \cdot n^2)$.

We proceed in three steps. The first step is to identify in $\mathcal{O}(p(k)^2 \cdot n^2)$ time a set $X \subseteq B$, $|X| \geq 16k^2 + 16k + 1$, such that if $\mathcal{I}$ is a YES instance, then for every $x \in X$ there is a solution η with $\mathcal{P} = \eta(E(H))$ having the following properties:

(1.i) at least $3k+1$ vertices of B are free from $\mathcal{P}$;

(1.ii) at least $2k$ arcs of the matching have both endpoints free from $\mathcal{P}$;

(1.iii) if x is accessed by some path $P \in \mathcal{P}$ from a vertex v, then $v \in A$.

The second step of the proof is to show that one can identify in $\mathcal{O}(p(k)^2 \cdot n^2)$ time a vertex $x \in X$ such that if $\mathcal{I}$ is a yes instance, then there is a solution with $\mathcal{P} = \eta(E(H))$ having the following properties:

(2.i) at least $k+1$ vertices of B are free from $\mathcal{P}$;

(2.ii) if x is accessed by some path $P \in \mathcal{P}$ from a vertex v, then $v \in A$;

(2.iii) if x is left by some path $P \in \mathcal{P}$ to a vertex v, then $v \in C$.

The final, concluding step of the proof is to show that there is a solution $\mathcal{P} = \eta(E(H))$ such that

(3.i) No path from $\mathcal{P}$ is using x.

We proceed with the first step. Let $\mathcal{Q} = \eta(E(H))$, where η is the solution for the Rooted Immersion instance with the minimum sum of lengths of the paths.

For every vertex $b \in B$, we identify two sets: G_b, R_b. The set R_b consists of those inneighbors of b outside A, which are inneighbors of at least $6k$ vertices from B, while G_b consists of the rest of inneighbors of b outside A. Formally,

$$R_b = \{v \mid v \in V(T) \setminus A \wedge (v,b) \in E \wedge |N^+(v) \cap B| \geq 6k\},$$
$$G_b = \{v \mid v \in V(T) \setminus A \wedge (v,b) \in E \wedge |N^+(v) \cap B| < 6k\}.$$

Note that R_b, G_b can be computed in $\mathcal{O}(n^2)$ time. Let $B_\emptyset$ be the set of those vertices $b \in B$, for which $G_b = \emptyset$. We claim that if $|B_\emptyset| \geq 16k^2 + 16k + 1$, then we can set $X = B_\emptyset$.

Take any $b \in B_\emptyset$. We argue that we can reroute the paths of $\mathcal{Q}$ that access b from outside A in such a manner, that during rerouting each of them we use at most one additional free vertex from B and at most one additional arc from the matching. We reroute the paths one by one. Take path P that accesses b from outside A, and let v be the previous vertex on the path. As $G_b = \emptyset$, $v \in R_b$. Therefore, v has at least $6k$ outneighbors in B. Out of them, at most $4k$ are not free with respect to $\mathcal{Q}$, due to Observation 9.6, while at most $k-1$ were used by previous reroutings. Therefore, there is a vertex $b' \in B \cap N^+(v)$, such that b' is still free. Thus we can substitute usage of the arc (v,b) on P by the path $v \to b' \to c \to a \to b$, where (c,a) is an arbitrary arc of the matching that still has both endpoints free, which exists due to using at most $k-1$ of them so far. After the rerouting we examine the obtained walk and remove all the cycles on this walk so that we obtain a simple path. Note that this shortcutting step cannot spoil property (1.iii) for this path.

We are now left with the case when $|B_\emptyset| < 16k^2 + 16k + 1$. Let $B_g = B \setminus B_\emptyset$. Then $|B_g| \geq 4(16k^2 + 16k + 1)$. We construct a semi-complete digraph $S = (B_g, L)$ as follows. For every $b_1, b_2 \in B_g$, $b_1 \neq b_2$, we put arc (b_1, b_2) if for every $v \in G_{b_1}$, either $v \in G_{b_1} \cap G_{b_2}$ or v has an outneighbor in G_{b_2}. Similarly, we put arc (b_2, b_1) into L if for every $v \in G_{b_2}$, either $v \in G_{b_1} \cap G_{b_2}$ or v has an outneighbor in G_{b_1}. By applying Lemma 9.8 to the semi-complete bipartite graph $((G_{b_1} \setminus G_{b_2}) \cup (G_{b_2} \setminus G_{b_1}), E(G_{b_1} \setminus G_{b_2}, G_{b_2} \setminus G_{b_1}))$ we infer that for every pair of distinct $b_1, b_2 \in B_g$ there is at least one arc with endpoints b_1 and b_2. Hence S is semi-complete. The definition of S gives raise to a straightforward algorithm constructing it in $\mathcal{O}(p(k)^2 \cdot n^2)$ time.

Let X be the set of vertices of B_g that have outdegree at least $6k^2+6k$ in S; note that X can be constructed in $\mathcal{O}(p(k)^2)$ time. Observe that $|X| \geq 16k^2+16k+1$, since otherwise the sum of the outdegrees in S would be at most $(16k^2+16k)(|B_g|-1) + (|B_g| - 16k^2 - 16k)(6k^2+6k-1)$, which is smaller than $\binom{|B_g|}{2}$ for $|B_g| \geq 4(16k^2+16k+1)$.

We now claim that for every $b \in X$, every path of $\mathcal{Q}$ using vertex b can be rerouted at the cost of using at most two free vertices of B and at most two arcs from the matching that have still both endpoints free. We perform reroutings one by one. Assume that there is a path $P \in \mathcal{Q}$ accessing b from outside A. Let v be the predecessor of b on P. If $v \in R_b$, then we use the same rerouting strategy as in the case of large $B_\emptyset$. Assume then that $v \in G_b$. As $b \in X$, its outdegree in S is at least $6k^2+6k$. This means that there are at least $6k^2+6k$ vertices b' in

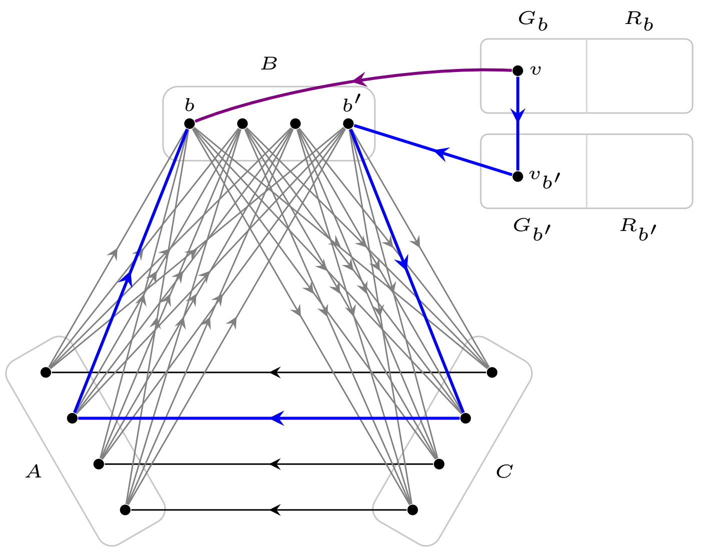

Figure 4: Rerouting strategy for a path accessing vertex b from G_b. The original path is depicted in violet, while the rerouted path is depicted in blue.

B_g and corresponding vertices $v_{b'} \in N^-(b')$, such that for every b' either $v_{b'} = v$ or $(v, v_{b'}) \in E$. Out of these $6k^2 + 6k$ vertices b', at most $4k$ are not free due to Observation 9.6, at most $2k - 2$ were used in previous reroutings, which leaves us with at least $6k^2$ vertices b' still being free. If for any such b' we have $v_{b'} = v$, we follow the same rerouting strategy as in the case of large $B_\emptyset$. Assume then that these $6k^2$ vertices $v_{b'}$ are all distinct from v; note that, however, they are not necessarily distinct from each other. As each $v_{b'}$ belongs to $G_{b'}$, $v_{b'}$ can have at most $6k - 1$ outneighbors in B. Hence, each vertex of $V(T)$ can occur among these $6k^2$ vertices $v_{b'}$ at most $6k - 1$ times, so we can distinguish at least $k + 1$ pairwise distinct vertices $v_{b'}$. We have that arcs $(v, v_{b'})$ and $(v_{b'}, b')$ exist, while b' is still free. By Observation 9.2, at most k arcs $(v, v_{b'})$ are used by some paths, which leaves us at least one $v_{b'}$, for which arc $(v, v_{b'})$ is free. We can now substitute the arc (v, b) in P by the path $v \to v_{b'} \to b' \to c \to a \to v$, where (c, a) is an arbitrarily chosen arc from the matching that still has both endpoints free, which exists due to using at most $2k - 2$ of them so far. See Figure 4. After rerouting, remove all cycles that appeared on the obtained walk in order to obtain a simple path; again, this shortcutting step does not spoil property (1.iii) for the path. Thus, Observation 9.2 still holds after rerouting. Observe that in this manner we use additional vertex b' that was free, additional one arc (c, a) from the matching, whereas passing the path through $v_{b'}$ can spoil at most one arc of the matching that still had both endpoints free, or at most one free vertex from B. This concludes the construction of the set X.

We proceed with the second step of the proof. We mimic the rerouting arguments from the first step to obtain a vertex $x \in X$ with the following property: the rerouted family of paths $\mathcal{P}$ obtained in the first step that can access x only from A, can be further rerouted so that every path can only leave x by accessing some vertex from C.

For every $b \in X$ consider sets R'_b and G'_b defined similarly as before:

$$R'_b = \{v \mid v \in V(T) \setminus C \wedge (b, v) \in E \wedge |N^-(v) \cap B| \geq 8k\},$$
$$G'_b = \{v \mid v \in V(T) \setminus C \wedge (b, v) \in E \wedge |N^-(v) \cap B| < 8k\}.$$

Assume first that there is some $y \in X$, such that $G'_y = \emptyset$. We argue that we can in such a case set $x = y$. Firstly, reroute a solution that minimizes the total sum of lengths of the paths obtaining a solution with the family of paths $\mathcal{P}$ that uses at most $2k$ additional free vertices from B and at most $2k$ additional arcs from the matching that had both endpoints free, but does not access x from outside A. One by one we reroute paths that traverse y. Each rerouting will cost at most one free vertex from B and at most one arc from the matching that has still both endpoints free. Let P be a path from the solution that passes through y and let $v \in R'_y$ be the next vertex on P. The vertex v has at least $8k$ inneighbors in B; at most $4k$ of them could be used by the original solution, at most $2k$ of them could be used in rerouting during the first phase and at most $k - 1$ of them could be used during previous reroutings in this phase. Therefore, we are left with at least one vertex $y' \in B$ that is still free, such that $(y', v) \in E(T)$. We can now substitute the arc (y, v) in P by the path $y \to c \to a \to y' \to v$, where (c, a) is an arbitrarily chosen arc from the matching that was not yet used, which exists due to using at most $3k-1$ of them so far. Again, we shortcut all the cycles that appeared after this substitution so that we obtain a simple path. Note that this shortcutting step spoils neither property (2.ii) nor (2.iii) for the path.

We are left with the case when G'_y is nonempty for all $y \in X$. Construct a digraph $S' = (X, L')$ in symmetrically to the previous construction: put arc (b_1, b_2) into L' iff for every $v_{b_2} \in G'_{b_2}$ there exists $v_{b_1} \in G'_{b_1}$ such that $v_{b_1} = v_{b_2}$ or $(v_{b_1}, v_{b_2}) \in E$. The remark after Lemma 9.8 ensures that S' is semi-complete. Again, S' can be computed in $\mathcal{O}(p(k)^2 \cdot n^2)$ time.

As $|X| \geq 16k^2 + 16k + 1$, there exists $x \in X$, which has indegree at least $8k^2 + 8k$ in S'; note that x can be found in $\mathcal{O}(p(k)^2)$ time. As before, we argue that after the first rerouting phase for x, we can additionally reroute the paths so that every path can leave x only into C. We reroute the paths one by one; each rerouting uses at most two free vertices from B and at most two arcs from the matching that still had both endpoints free. As the indegree of x in S' is at least $8k^2 + 8k$, we have at least $8k^2 + 8k$ vertices $x' \in X$ and corresponding $v_{x'} \in G'_{x'}$, such that $v_{x'} = v$ or $(v_{x'}, v) \in E$. At most $4k$ of them were used in $\mathcal{Q}$, at most $2k$ were used in the first phase of rerouting, and at most $2k - 2$ of them were used in this phase of rerouting. This leaves at least $8k^2$ vertices x' which are still free. If for any of them we have $v_{x'} = v$, we can make the rerouting similarly as in the previous case: we substitute the arc (x, v) with the path $x \to c \to a \to x' \to v$, where (c, a) is an arbitrary arc of the matching that still has both endpoints free, which exists due to using at most $4k - 2$ of them so far. Assume then, that all vertices $v_{x'}$ are distinct from v; note that, however, they are not necessarily distinct from each other. As for every x' we have $v_{x'} \in G'_{x'}$, by the definition of $G'_{x'}$ the vertices $v_{x'}$ can have at most $8k - 1$ inneighbors in X. This means that every vertex of $V(T)$ can occur among vertices $v_{x'}$ at most $8k - 1$ times, which proves that there are at least $k + 1$ pairwise distinct vertices among them. By Observation 9.2, for at most k of them the arc outgoing to v can be already used, which leaves us with a single vertex x' such that arcs $(x', v_{x'})$ and $(v_{x'}, v)$ exist and are not yet used, whereas x' is still free. Now we can perform the rerouting as follows: we substitute the arc (x, v) in P by the path $x \to c \to a \to x' \to v_{x'} \to v$, where (c, a) is an arbitrary arc from the matching that still had both endpoints free. Such an arc exists since we used at most $2k$ such arcs in the first phase of the rerouting, and at most $2k - 2$ in this phase of the rerouting. Similarly as before, we use one additional free vertex x' from B, one additional arc (c, a) from the matching, while usage of $v_{x'}$ can spoil at most one free vertex from B or at most one arc from the matching. After this, we delete all the possible cycles created on the path in order to make Observation 9.2 still hold; again, this shortcutting step spoils neither property (2.ii) nor property (2.iii). This concludes the construction of the vertex x.

To finish the proof it remains to show that after performing the two phases of rerouting and obtaining a solution η', whose paths can access and leave x only from A and into C, we can reroute every path so that it does not traverse x at all. Note that so far we have used at most

$4k$ vertices from B, so we still have at least $k+1$ vertices unused. Observe that at most k of these vertices can belong to $\eta'(V(H))$, which leaves us with at least one vertex x' that is still free and is not an image of any vertex of H in η'.

If $x \in \eta(u)$ for some $u \in V(H)$, then we simply move the image u: we consider η'' that differs from η' by replacing x with x' both in the image of u and in all the paths from $\eta'(E(H))$ which traverse x. Note that we can make this move since x is not a root vertex: the triple does not contain any roots. In case when $x \notin \eta(V(H))$ we can perform the same rerouting scheme: in all the paths from $\eta'(E(H))$ we substitute every appearance of x with x'. Then no path traverses x, so $\mathcal{I}' = ((H, u_1, u_2, \ldots, u_t), (T \setminus \{x\}, v_1, v_2, \ldots, v_t))$ is a YES instance if $\mathcal{I}$ was. $\square$

9.2 Applying the irrelevant vertex rule

Armed with the irrelevant vertex rule, we can proceed to the algorithm for Rooted Immersion.

Theorem 9.9. *There exists an algorithm that, given a rooted semi-complete digraph* $\mathbf{T}$ *on* n *vertices and a rooted digraph* $\mathbf{H}$*, in time* $f(|H|) \cdot n^3$ *checks whether there exists a rooted immersion from* $\mathbf{H}$ *to* $\mathbf{T}$*, for some elementary function* f*.*

Proof. Let f be the function given by Lemma 3.6; i.e., basing on a $f(t)$-jungle in any semi-complete digraph S, one can find a t-triple in S in time $\mathcal{O}(|V(S)|^3 \log |V(S)|)$. Moreover, let p be the polynomial given by Theorem 9.1; i.e., in a $p(|H|)$-triple that is disjoint from the roots one can find an irrelevant vertex for the Rooted Immersion problem in $\mathcal{O}(p(|H|)^2 \cdot n^2)$ time.

Given the input semi-complete rooted digraph T and a rooted digraph H, we run the approximation algorithm of Theorem 4.12 for parameters $5k$ and $130k$ on T with the roots removed, where $k = f(p(|H|))$; this takes at most $g(|H|) \cdot n^2$ time for some elementary function g. If the algorithm returns a decomposition of T without roots of width at most $140k$, we include all the roots in every bag of the decomposition and finalize the algorithm by running the dynamic programming routine for Rooted Immersion (Theorem 6.3), which takes $h(|H|) \cdot n$ time for some elementary function h. Otherwise, using Lemma 3.9 or Lemma 3.12 we extract a $(k, 4)$-short jungle X from the output $(130k + 2, 5k)$-degree tangle or $(5k + 1, 5k)$-matching tangle; this takes $\mathcal{O}(k^3 \cdot n^2)$ time.

Obviously, X is also a k-jungle in the sense of Fradkin and Seymour, so we are tempted to run the algorithm of Lemma 3.6 to extract a triple; however, the running time is a bit too much. We circumvent this obstacle in the following manner. As X is a $(k, 4)$-short jungle, then if we define S to be the subdigraph induced in T by X and, for every pair v, w of vertices in X, k internally vertex-disjoint paths of length at most 4 from v to w, then X is still a $(k, 4)$-short jungle in S, but S has size polynomial in k. As we store the short jungle together with the corresponding family of paths between the vertices, we can construct S in $\mathcal{O}(k^{\mathcal{O}(1)})$ time and, using Lemma 3.6, in $\mathcal{O}(k^{\mathcal{O}(1)})$ time find a $p(|H|)$-triple inside S. This $p(|H|)$-triple is of course also a $p(|H|)$-triple in T. We apply Theorem 9.1 to find an irrelevant vertex in this triple in $p(|H|)^2 \cdot n^2$ time, delete it, and restart the algorithm.

Since there are n vertices in the graph, the algorithm makes at most n iterations. Since every iteration takes $h(k) \cdot n^2$ time for some elementary function h, the claim follows. $\square$

10 Dynamic programming routines for containment relations

In this section we provide details of the dynamic programming routines that solve containment problems when a path decomposition of small width is given. First, we explain the terminology used to describe the constructed parts of expansions or immersions. Then we explain the routine for topological containment that is somewhat simpler, and finally proceed to immersion. But before all of these, let us give some intuition behind the algorithms.

The main idea of our routine is as follows: we will encode the interaction of the model of H with all the already introduced vertices as sequences of paths in a standard folio manner. Every such path has to end in the separator, but can begin in any forgotten vertex since it may be accessed from a vertex not yet introduced. In general setting the dynamic program would need to remember this first vertex in order to check whether it can be indeed accessed; that would yield an XP algorithm and, in essence, this is exactly the idea behind the algorithm of Fradkin and Seymour [31]. However, if the digraph is semi-complete, then between every not yet introduced vertex and every forgotten vertex there is an arc. Therefore, we do not need to remember the forgotten vertex itself to check accessibility; a marker saying *forgotten*, together with information about which markers in fact represent the same vertices in case of immersion, will suffice.

We hope that a reader well-familiar with construction of dynamic programs on various decompositions already has a crude idea about how the computation will be performed. Let us proceed with the details in the next subsections.

10.1 Terminology

First, we need to introduce definitions that will enable us to encode all the possible interactions between a model of a digraph H and a separation. Let (A, B) be a separation of T, where T is a given semi-complete digraph.

In the definitions we use two special symbols: $\mathfrak{F}, \mathfrak{U}$; the reader can think of them as an arbitrary element of $A \setminus B$ (*forgotten*) and $B \setminus A$ (*unknown*), respectively. Let $\iota : V(T) \to (A \cap B) \cup \{\mathfrak{F}, \mathfrak{U}\}$ be defined as follows: $\iota(v) = v$ if $v \in A \cap B$, whereas $\iota(v) = \mathfrak{F}$ for $v \in A \setminus B$ and $\iota(v) = \mathfrak{U}$ for $v \in B \setminus A$.

Definition 10.1. *Let P be a path. A sequence of paths $(P_1, P_2, \ldots, P_h)$ is a* trace *of P with respect to (A, B), if P_i for $1 \leq i \leq h$ are all maximal subpaths of P that are fully contained in $T[A]$, and the indices in the sequence reflect their order on path P.*

Let $(P_1, P_2, \ldots, P_h)$ be the trace of P with respect to (A, B). A signature *of P on (A, B) is a sequence of pairs $((b_1, e_1), (b_2, e_2), \ldots, (b_h, e_h))$, where $b_h, e_h \in (A \cap B) \cup \{\mathfrak{F}\}$, such that for every $i \in \{1, 2, \ldots, h\}$:*

- *b_i is the beginning of path P_i if $b_i \in A \cap B$, and $\mathfrak{F}$ otherwise;*
- *e_i is the end of path P_i if $e_i \in A \cap B$, and $\mathfrak{F}$ otherwise.*

In other words, b_i, e_i are images of the beginning and the end of path P_i in mapping ι. Observe the following properties of the introduced notion:

- Signature of a path P on separation (A, B) depends only on its trace; therefore, we can also consider signatures of traces.
- It can happen that $b_i = e_i \neq \mathfrak{F}$ only if P_i consists of only one vertex $b_i = e_i$.
- From the definition of separation it follows that only for $i = h$ it can happen that $e_i = \mathfrak{F}$, since there is no arc from $A \setminus B$ to $B \setminus A$.
- The empty signature corresponds to P entirely contained in $B \setminus A$.

Now we are able to encode relevant information about a given expansion of H.

Definition 10.2. *Let η be an expansion of a digraph H in T. An* expansion signature *of η on (A, B) is a mapping ρ such that:*

- *for every $v \in V(H)$, $\rho(v) = \iota(\eta(v))$;*
- *for every $a \in E(H)$, $\rho(a)$ is a signature of $\eta(a)$ on (A, B).*

The set of possible expansion signatures on separation (A, B) will be denoted by $\mathcal{V}_{(A,B)}$. Observe that in an expansion η all the paths in $\eta(E(H))$ are internally vertex-disjoint, so the non-forgotten beginnings and ends in all the signatures of paths from $\rho(E(H))$ can be equal only if they are in the same pair or correspond to a vertex from $\rho(V(H))$ at the beginning or at the end of a signature of some path. Armed with this observation, we can bound the number of possible expansion signatures on a separation of small order.

Lemma 10.3. *If $|V(H)| = k, |E(H)| = \ell, |A \cap B| = m$, then the number of possible different expansion signatures on (A, B) is at most*

$$(m+2)^k \cdot (m+2)^m \cdot m^\ell \cdot m! \cdot (m+2)^\ell \cdot (m+2)^\ell = 2^{\mathcal{O}((k+\ell+m)\log m)}.$$

Moreover, all of them can be enumerated in $2^{\mathcal{O}((k+\ell+m)\log m)}$ time.

Proof. The consecutive terms correspond to:

1. the choice of mapping ρ on $V(H)$;
2. for every element of $(A \cap B) \setminus \rho(V(H))$, choice whether it will be the end of some subpath in some path signature, and in this case, the value of corresponding beginning (a vertex from $A \cap B$ or $\mathfrak{F}$);
3. for every pair composed in such manner, choice to which $\rho(a)$ it will belong;
4. the ordering of pairs along the path signatures;
5. for every $(v, w) \in E(H)$, choice whether to append a pair of form $(b, \rho(w))$ at the end of the signature $\rho((v, w))$, and in this case, the value of b (a vertex from $A \cap B$ or $\mathfrak{F}$).
6. for every $(v, w) \in E(H)$, choice whether to append a pair of form $(\rho(v), e)$ at the beginning of the signature $\rho((v, w))$, and in this case, the value of e (a vertex from $A \cap B$ or $\mathfrak{F}$).

It is easy to check that using all these information one can reconstruct the whole signature. For every object constructed in the manner above we can check in time polynomial in k, ℓ, m, whether it corresponds to a possible signature. This yields the enumeration algorithm. $\square$

We now proceed to encoding intersection of an immersion with a given separation (A, B). Unfortunately, the definition must be slightly more complicated for the following reason. Assume that we have two subpaths P_1, P_2 in some path traces that both start in some vertices b_1, b_2 that are forgotten, i.e., $b_1, b_2 \in A \backslash B$. Observe that not only need to remember that $\iota(b_1) = \iota(b_2) = \mathfrak{F}$, but also need to store the information whether $b_1 = b_2$: in the future computation we might need to know whether b_1 and b_2 are actually not the same vertex, in order to prevent using twice the same arc incoming to this vertex from the unknown region, in two different images of paths. Fortunately, this is the only complication.

Definition 10.4. *Let η be an immersion of a digraph H in T. An* immersion signature *of η on (A, B) is a mapping ρ together with an equivalence relation $\equiv$ on the set of all the pairs of form $(\mathfrak{F}, e)$ appearing in the image of ρ, such that:*

- *for every $v \in V(H)$, $\rho(v) = \iota(\eta(v))$;*
- *for every $a \in E(H)$, $\rho(a)$ is a signature of $\eta(a)$;*
- $(\mathfrak{F}, e_1) \equiv (\mathfrak{F}, e_2)$ *if and only if markers $\mathfrak{F}$ in both pairs correspond to the same forgotten vertex before being mapped by ι.*

We remark that the same pair of form $(\mathfrak{F}, e)$ can appear in different signatures; in this case, by somehow abusing the notation, we consider all the appearances as different pairs. Also, we often treat the equivalence relation $\equiv$ as part of the mapping ρ, thus denoting the whole signature by ρ. We denote the set of possible immersion signatures on separation (A, B) by $\mathcal{E}_{(A,B)}$. Similarly as in Lemma 10.3, we can bound the number of immersion signatures on a separation of small order.

Lemma 10.5. *If $|V(H)| = k, |E(H)| = \ell, |A \cap B| = m$, then the number of possible different immersion signatures on (A, B) is bounded by*

$$(m+2)^k \cdot ((m+2)^m \cdot m! \cdot (m+2)^2)^\ell \cdot B_{(m+2)\ell} = 2^{\mathcal{O}(k \log m + m\ell(\log \ell + \log m))}.$$

Moreover, all of them can be enumerated in $2^{\mathcal{O}(k \log m + m\ell(\log \ell + \log m))}$ time.

Proof. The consecutive terms correspond to:

1. the choice of mapping ρ on $V(H)$;
2. for every arc $a = (v, w) \in E(H)$ the complete information about the signature $\rho(a)$:
   - for every element of $A \cap B$, whether it will be the end of some path in the signature, and in this case, the value of corresponding beginning (a vertex from $A \cap B$ or $\mathfrak{F}$),
   - the ordering of pairs along the signature,
   - whether to append a pair of form $(b, \rho(w))$ at the end of the signature $\rho(a)$, and in this case, the value of b (a vertex from $A \cap B$ or $\mathfrak{F}$),
   - whether to append a pair of form $(\rho(v), e)$ at the beginning of the signature $\rho(a)$, and in this case, the value of e (a vertex from $A \cap B$ or $\mathfrak{F}$).
3. partition of at most $(m+2)l$ pairs in all the signatures from $\rho(E(H))$ into equivalence classes with respect to $\equiv$.

In the last term we used Bell numbers B_n, for which a trivial bound $B_n \leq n^n$ applies.

It is easy to check that using all these information one can reconstruct the whole signature. For every object constructed in the manner above we can check in time polynomial in k, l, m, whether it corresponds to a possible signature. This yields the enumeration algorithm. □

10.2 The algorithm for topological containment

Finally, we are able to present the dynamic programming routine for topological containment. Recall that we will solve a slightly more general problem, where arcs from some subset $F \subseteq E(H)$, called constraints, are required to be mapped to single arcs instead of possibly longer paths. We first prove a simple lemma that will be useful to handle the constraints.

Lemma 10.6. *Let H, G be simple digraphs, and let $F \subseteq E(H)$. Then H can be topologically embedded in G with constraints F if and only if $H' = H \setminus F$ can be topologically embedded in G using expansion η such that $(\eta(v), \eta(w)) \in E(G)$ for every $(v, w) \in F$.*

Proof. From left to right, if η is an expansion of H in G that respects constraints F, then η restricted to H' is also an expansion of H' in G, and moreover $(\eta(v), \eta(w)) \in E(G)$ for every $(v, w) \in F$ since η respects constraints F. From right to left, assume that η is an expansion of H' in G such that $(\eta(v), \eta(w)) \in E(G)$ for every $(v, w) \in F$. Then we can extend η to H by setting $\eta((v, w)) = (\eta(v), \eta(w))$ for every $(v, w) \in F$, and these new paths will be pairwise internally vertex-disjoint, and internally vertex-disjoint from $\eta(a)$ for $a \notin F$. $\square$

We are ready to provide the dynamic programming routine.

Theorem 10.7 (Theorem 6.2, restated). *There exists an algorithm that, given a digraph H with k vertices, ℓ arcs, a set $F \subseteq E(H)$ of constraints, and a semi-complete digraph T on n vertices together with its path decomposition of width p, checks whether H is topologically contained in T with constraints F in time $2^{\mathcal{O}((p+k+\ell)\log p)} \cdot n$.*

Proof. Let $H' = H \setminus F$, and let $W = (W_1, \ldots, W_r)$ be a path decomposition of T of width p. Without loss of generality we assume that W is a nice path decomposition and $r = \mathcal{O}(n)$.

By Lemma 10.3, for every separation $(A, B) = (\bigcup_{j=1}^{i} W_j, \bigcup_{j=i}^{r} W_j)$ with separator W_i the number of possible signatures is $2^{\mathcal{O}((p+k+\ell)\log p)}$. We will consecutively compute the values of a binary table $D_{(A,B)} : \mathcal{V}_{(A,B)} \to \{\bot, \top\}$ with the following meaning. For $\rho \in \mathcal{V}_{(A,B)}$, $D_{(A,B)}[\rho]$ tells whether there exists a mapping $\overline{\rho}$ with the following properties:

- for every $v \in V(H)$, $\overline{\rho}(v) = \rho(v)$ if $\rho(v) \in (A \cap B) \cup \{\mathfrak{U}\}$ and $\overline{\rho}(v) \in A \setminus B$ if $\rho(v) = \mathfrak{F}$;
- for every $a = (v, w) \in E(H')$, $\overline{\rho}(a)$ is a correct path trace with signature $\rho(a)$, beginning in $\overline{\rho}(v)$ if $\overline{\rho}(v) \in A$ and anywhere in A otherwise, ending in $\overline{\rho}(w)$ if $\overline{\rho}(w) \in A$ and anywhere in $A \cap B$ otherwise;
- path traces $\overline{\rho}(a)$ are vertex-disjoint, apart possibly from meeting at the ends if the ends correspond to images of appropriate endpoints of arcs in $\overline{\rho}$;
- for every $a = (v, w) \in F$, if $\overline{\rho}(v) \neq \mathfrak{U}$ and $\overline{\rho}(w) \neq \mathfrak{U}$, then $(\overline{\rho}(v), \overline{\rho}(w)) \in E(T)$.

Such mapping $\overline{\rho}$ will be called a *partial expansion of H' on (A, B) respecting constraints F*. Note that we may also talk about signatures of partial expansions; in the definition above, ρ is the signature of $\overline{\rho}$ on (A, B).

For the first separation $(\emptyset, V(T))$ we have exactly one signature with value $\top$, being the signature which maps all the vertices into $\mathfrak{U}$ and all the arcs into empty signatures. By Lemma 10.6, the result of the whole computation should be the value for the signature on the last separation $(V(T), \emptyset)$ which maps all vertices into $\mathfrak{F}$ and arcs into signatures consisting of one pair $(\mathfrak{F}, \mathfrak{F})$. Therefore, it suffices to show how to fill the values of the table for **introduce vertex** step and **forget vertex** step. Note that the introduce bags of the decomposition may be viewed as introducing a vertex v to a separation (A, B), i.e., considering the next separation $(A \cup \{v\}, B)$ for $v \notin A$, $v \in B$. Similarly, forget bags may be viewed as forgetting a vertex w from a separation $(A, B \cup \{w\})$ with $w \in A$, $w \notin B$, i.e., considering the next separation (A, B).

Introduce vertex step. Let us introduce vertex $v \in B \setminus A$ to the separation (A, B), i.e., we consider the new separation $(A \cup \{v\}, B)$. Let $\rho \in \mathcal{V}_{(A \cup \{v\}, B)}$. We show that $D_{(A \cup \{v\}, B)}[\rho]$ can be computed using the stored values of $D_{(A,B)}$ by analyzing the way signature ρ interferes with vertex v. In each case we argue that one can take $D_{(A \cup \{v\}, B)}[\rho] = \bigvee_{\rho' \in \mathcal{G}} D_{(A,B)}[\rho']$ for some set $\mathcal{G}$ that corresponds to possible trimmings of ρ to the previous separation of smaller order. Formally, one needs to argue that (i) if there exists a partial expansion with signature ρ on $(A \cup \{v\}, B)$, then after deleting vertex v this partial expansion has some signature $\rho' \in \mathcal{G}$ on

(A, B), and (ii) if there exists a partial expansion with signature ρ' on (A, B) such that $\rho' \in \mathcal{G}$, then one can extend this partial expansion using vertex v to obtain a partial expansion on $(A \cup \{v\}, B)$ with signature ρ. Since this check in each case follows directly from the explanations provided along with the construction of $\mathcal{G}$, we leave the formal verification of this statement to the reader.

Case 1*:* $v \notin \rho(V(H))$, that is, v is not an image of a vertex of H. Observe that in this situation v can be contained in at most one pair of at most one signature $\rho(a)$ for some $a \in E(H')$.

Case 1.1*:* $b_i = v = e_i$ for some pair $(b_i, e_i) \in \rho(a)$ and some $a \in E(H')$. This means that the signature of the partial expansion truncated to separation (A, B) must look exactly like ρ, but without this subpath of length zero. Thus $D_{(A\cup\{v\},B)}[\rho] = D_{(A,B)}[\rho']$, where ρ' is constructed from ρ by deleting this pair from the corresponding signature.

Case 1.2*:* $b_i = v \neq e_i$ for some pair $(b_i, e_i) \in \rho(a)$ and some $a \in E(H')$. This means that the partial expansion truncated to separation (A, B) has to look the same but for the path corresponding to this very pair, which needs to be truncated by vertex v. The new beginning has to be either a vertex in $A \cap B$, or a forgotten vertex from $A \setminus B$. As T is semi-complete and (A, B) is a separation, there is an arc from v to every vertex of $A \setminus B$. Therefore, $D_{(A\cup\{v\},B)}[\rho] = \bigvee_{\rho'} D_{(A,B)}[\rho']$, where the disjunction is taken over all signatures ρ' differing from ρ as follows: in ρ' the pair (b_i, e_i) is substituted with (b'_i, e_i), where $b'_i = \mathfrak{F}$ or b'_i is any vertex of $A \cap B$ such that there is an arc (v, b'_i).

Case 1.3*:* $b_i \neq v = e_i$ for some pair $(b_i, e_i) \in \rho(a)$ and some $a \in E(H')$. Similarly as before, the partial expansion truncated to separation (A, B) has to look the same but for the path corresponding to this very pair, which needs to be truncated by vertex v. As (A, B) is a separation, the previous vertex on the path has to be in the separator $A \cap B$. Therefore, $D_{(A\cup\{v\},B)}[\rho] = \bigvee_{\rho'} D_{(A,B)}[\rho']$, where the disjunction is taken over all signatures ρ' differing from ρ as follows: in ρ' the pair (b_i, e_i) is substituted with (b_i, e'_i), where e'_i is any vertex of $A \cap B$ such that there is an arc (e'_i, v).

Case 1.4*:* v is not contained in any pair in any signature from $\rho(E(H'))$. Either v lies on some path in the partial expansion, or it does not. In the first case the corresponding path in partial expansion on $(A \cup \{v\}, B)$ has to be split into two subpaths, when truncating the expansion to (A, B). Along this path, the arc that was used to access v had to come from inside $A \cap B$, due to (A, B) being a separation; however, the arc used to leave v can go to $A \cap B$ or to any forgotten vertex from $A \setminus B$, as (A, B) is a separation and T is a semi-complete digraph. In the second case, the signature of the truncated expansion stays the same. Therefore, $D_{(A\cup\{v\},B)}[\rho] = D_{(A,B)}[\rho] \vee \bigvee_{\rho'} D_{(A,B)}[\rho']$, where the disjunction is taken over all signatures ρ' differing from ρ as follows: in ρ' exactly one pair (b_i, e_i) is substituted with two pairs (b_i, e'_i) and (b'_i, e_i), where $e'_i \in A \cap B$ with $(e'_i, v) \in E(T)$, whereas $b'_i = \mathfrak{F}$ or $b'_i \in A \cap B$ with $(v, b'_i) \in E(T)$.

Case 2*:* $v = \rho(u)$ for some $u \in V(H)$. For every $(u, u') \in E(H')$, v has to be the beginning of the first pair of $\rho((u, u'))$; otherwise, $D_{(A\cup\{v\},B)}[\rho] = \bot$. Similarly, for every $(u', u) \in E(H')$, v has to be the end of the last pair of $\rho((u', u))$; otherwise, $D_{(A\cup\{v\},B)}[\rho] = \bot$. Furthermore, for every $(u, u') \in F$ such that $\rho(u') \neq \mathfrak{U}$, if $\rho(u') \in A \cap B$ then $(\rho(u), \rho(u'))$ must be an arc of T; otherwise, $D_{(A\cup\{v\},B)}[\rho] = \bot$. Finally, for every $(u', u) \in F$ such that $\rho(u') \neq \mathfrak{U}$, it must hold that $\rho(u') \in A \cap B$ and $(\rho(u'), \rho(u))$ must be an arc of T; otherwise, $D_{(A\cup\{v\},B)}[\rho] = \bot$. Assume then that all these four assertions hold. Then $D_{(A\cup\{v\},B)}[\rho] = \bigvee_{\rho'} D_{(A,B)}[\rho']$, where the disjunction is taken over all signatures ρ' such that: (i) ρ' differs on $V(H)$ from ρ only by having $\rho'(u) = \mathfrak{U}$; (ii) the first pairs of all

$\rho((u,u'))$ are truncated as in Case 1.2 for all $(u,u') \in E(H)$ (or as in Case 1.1, if the beginning and the end coincide), and (iii) the last pairs of all $\rho((u',u))$ are truncated as in Case 1.3 for all $(u',u) \in E(H)$ (or as in Case 1.1, if the beginning and the end coincide).

Forget vertex step Let us forget vertex $w \in A \setminus B$ from the separation $(A, B \cup \{w\})$, i.e., we consider the new separation (A,B). Let $\rho \in \mathcal{V}_{(A,B)}$; we argue that $D_{(A,B)}[\rho] = \bigvee_{\rho' \in \mathcal{G}} D_{(A \cup \{w\},B)}[\rho']$ for some set $\mathcal{G} \subseteq \mathcal{V}_{(A\cup\{w\},B)}$, which corresponds to possible extensions of ρ to the previous separation of larger order. Formally, one needs to argue that (i) if there exists a partial expansion with signature ρ on (A,B), then this partial expansion has signature $\rho' \in \mathcal{G}$ on $(A, B \cup \{w\})$, and (ii) if there exists a partial expansion with signature ρ' on $(A, B \cup \{w\})$ such that $\rho' \in \mathcal{G}$, then the signature of this partial expansion on (A,B) is ρ. Since this check in each case follows directly from the explanations provided along with the construction of $\mathcal{G}$, we leave the formal verification of this statement to the reader.

We now discuss, which signatures ρ' are needed in $\mathcal{G}$ by considering all the signatures $\rho' \in \mathcal{V}_{(A\cup\{w\},B)}$ partitioned with respect to behaviour on vertex w. For a fixed signature ρ' we put constraints on how ρ' must look like to be included in $\mathcal{G}$: we first put a constraint on the image of $V(H)$ in ρ', and then for every $a \in E(H')$ we list the possible values of $\rho'(a)$. In $\mathcal{G}$ we take into the account all signatures ρ' that (i) satisfy the imposed constraints on the image of $V(H)$, (ii) have one of the correct values for every $a \in E(H')$.

Case 1*:* $w \notin \rho'(V(H))$, that is, w is not in the image of $V(H)$. In this case we require that $\rho'|_{V(H)} = \rho|_{V(H)}$. We now inspect one arc $a \in E(H')$ and determine the correct values of $\rho'(a)$ by looking at all possible values of $\rho'(a)$ partitioned with respect to behaviour on w.

Case 1.1*:* $b_i = w = e_i$ for some pair $(b_i, e_i) \in \rho'(a)$. This means that in the corresponding partial expansions w had to be left to $B \setminus A$; however, in T there is no arc from w to $B \setminus A$ since (A,B) is a separation. Therefore, in $\mathcal{G}$ we consider no signatures ρ' having such a behaviour on any arc a.

Case 1.2*:* $b_i = w \neq e_i$ for some pair $(b_i, e_i) \in \rho'(a)$. If we are to consider ρ' in $\mathcal{G}$, then w must prolong some path from the signature ρ in such a manner that w is its beginning. After forgetting w the beginning of this path belongs to the forgotten vertices; therefore, in $\mathcal{G}$ we consider only signatures with $\rho'(a)$ differing from $\rho(a)$ on exactly one pair: in $\rho'(a)$ there is (w, e_i) instead of $(\mathfrak{F}, e_i)$ in $\rho(a)$.

Case 1.3*:* $b_i \neq w = e_i$ for some pair $(b_i, e_i) \in \rho'(a)$. This means that in the corresponding partial expansions w have to be left to $B \setminus A$; however, in T there is no arc from w to $B \setminus A$ since (A,B) is a separation. We obtain a contradiction; therefore, in $\mathcal{G}$ we consider no signatures having such a behaviour on any arc a.

Case 1.4*:* w is not contained in any pair of $\rho'(a)$. In this case, in the corresponding partial expansions w has to be either unused by the trace of a, or be an internal vertex of a path in this trace. In both cases the signatures on $(A, B \cup \{w\})$ and on (A,B) are equal. It follows that in $\mathcal{G}$ we can consider only signatures with $\rho'(a) = \rho(a)$ for such arcs a.

Case 2*:* $w = \rho'(u)$ for some $u \in V(H)$. We consider in $\mathcal{G}$ signatures ρ' that differ from ρ in following manner: (i) ρ' differs on $V(H)$ from ρ only by having $\rho'(u) = w$ for exactly one vertex $u \in V(H)$ whereas $\rho(u) = \mathfrak{F}$; (ii) for all arcs $(u,u') \in E(H')$, the first pair of $\rho'((u,u'))$ is of form (w, e_1), whereas the first pair of $\rho((u,u'))$ is of form $(\mathfrak{F}, e_1)$; (iii) for all arcs $(u',u) \in E(H')$, the last pair of $\rho'((u',u))$ is of form (b_h, w), whereas the last pair of $\rho((u',u))$ is of form $(b_h, \mathfrak{F})$ (or $(\mathfrak{F}, \mathfrak{F})$ in case $b_h = w$).

Updating table $D_{(A,B)}$ for each separation requires at most $\mathcal{O}(|\mathcal{V}_{(A,B)}|^2 \cdot (p+k+\ell)^{\mathcal{O}(1)}) = 2^{\mathcal{O}((p+k+\ell)\log p)}$ time, and since the number of separations in the path decomposition W is $\mathcal{O}(n)$, the theorem follows. $\square$

10.3 The algorithm for Rooted Immersion

Theorem 10.8 (Theorem 6.3, restated)**.** *There exists an algorithm that, given a rooted digraph $(H; v_1, v_2, \ldots, v_t)$ with k vertices and ℓ arcs and a semi-complete rooted digraph $(T; w_1, w_2, \ldots, w_t)$ on n vertices together with its path decomposition of width p, checks whether H can be immersed into T while preserving roots in time $2^{\mathcal{O}(k\log p + p\ell(\log \ell + \log p))} \cdot n$.*

Proof. Let $W = (W_1, \ldots, W_r)$ be the given path decomposition of T of width p. Without loss of generality we assume that W is a nice path decomposition and $r = \mathcal{O}(n)$.

By Lemma 10.5, for every separation $(A,B) = (\bigcup_{j=1}^{i} W_j, \bigcup_{j=i}^{r} W_j)$ with separator W_i we can bound the number of possible signatures by $2^{\mathcal{O}(k\log p + p\ell(\log \ell + \log p))}$. We show how to compute the values of a binary table $D_{(A,B)} : \mathcal{E}_{(A,B)} \to \{\bot, \top\}$ with the following meaning. For $\rho \in \mathcal{E}_{(A,B)}$, $D_{(A,B)}[\rho]$ tells, whether there exists a mapping $\overline{\rho}$ with the following properties:

- for every $v \in V(H)$, $\overline{\rho}(v) = \rho(v)$ if $\rho(v) \in (A \cap B) \cup \{\mathfrak{U}\}$ and $\overline{\rho}(v) \in A \setminus B$ if $\rho(v) = \mathfrak{F}$;
- for every $i = 1, 2, \ldots, t$, $\overline{\rho}(v_i) = w_i$ if $w_i \in A$ and $\overline{\rho}(v_i) = \mathfrak{U}$ otherwise;
- for every $a = (v,w) \in E(H)$, $\overline{\rho}(a)$ is a correct path trace with signature $\rho(a)$, beginning in $\overline{\rho}(v)$ if $\overline{\rho}(v) \in A$ and anywhere in A otherwise, ending in $\overline{\rho}(w)$ if $\overline{\rho}(w) \in A$ and anywhere in $A \cap B$ otherwise;
- all the paths in path traces $\overline{\rho}(a)$ are arc-disjoint for $a \in E(H)$.

Such mapping $\overline{\rho}$ will be called a *partial immersion of H on* (A,B). Note that we may also talk about signatures of partial immersions; in the definition above, ρ is the signature of $\overline{\rho}$ on (A,B).

For the first separation $(\emptyset, V(T))$ we have exactly one signature with value $\top$, being the signature which maps all the vertices into $\mathfrak{U}$ and all the arcs into empty signatures. The result of the whole computation should be the value for the signature on the last separation $(V(T), \emptyset)$, which maps all the vertices to $\mathfrak{F}$ and arcs to signatures consisting of one pair $(\mathfrak{F}, \mathfrak{F})$. Therefore, it suffices to show how to fill the values of the table for **introduce vertex** step and **forget vertex** step. Similarly as in Theorem 10.7, we view these steps as introducing and forgetting a vertex from a separation.

Introduce vertex step Let us introduce vertex $v \in B \setminus A$ to the separation (A,B), i.e., we consider the new separation $(A \cup \{v\}, B)$. Let $\rho \in \mathcal{E}_{(A\cup\{v\},B)}$, we need to show how to compute $D_{(A\cup\{v\},B)}[\rho]$ by a careful case study of how the signature ρ interferes with vertex v. If $v = w_i$ for some $i \in \{1, 2, \ldots, t\}$, then we consider only such ρ for which $\rho(v_i) = v$; for all the others we fill false values. Let us fix one $\rho \in \mathcal{E}_{(A\cup\{v\},B)}$. We argue that $D_{(A\cup\{v\},B)}[\rho] = \bigvee_{\rho' \in \mathcal{G}} D_{(A,B)}[\rho']$ for some set $\mathcal{G}$ that corresponds to possible trimmings of ρ to the previous separation of smaller order. Formally, one needs to argue that (i) if there exists a partial immersion with signature ρ on $(A \cup \{v\}, B)$, then after deleting vertex v this partial immersion has some signature $\rho' \in \mathcal{G}$ on (A,B), and (ii) if there exists a partial immersion with signature ρ' on (A,B) such that $\rho' \in \mathcal{G}$, then one can extend this partial immersion using vertex v to obtain a partial immersion on $(A \cup \{v\}, B)$ with signature ρ. Since this check in each case follows directly from the explanations provided along with the construction of $\mathcal{G}$, we leave the formal verification of this statement to the reader.

We examine every signature $\rho' \in \mathcal{E}_{(A,B)}$ and put constraints on how ρ' must look like to be included in $\mathcal{G}$: we first put a constraint on the image of $V(H)$ in ρ', and then for every $a \in E(H)$ we list the possible values of $\rho'(a)$. The set $\mathcal{G}$ consists of all the signatures ρ' that (i) satisfy the imposed constraints on the image of $V(H)$, (ii) have one of the correct values for every $a \in E(H)$, and (iii) satisfy some additional, global constraints that are described later.

Case 1*:* $v \notin \rho(V(H))$, that is, v is not being mapped on by any vertex of H. In this case we require that $\rho'|_{V(H)} = \rho|_{V(H)}$. We now examine one arc $a \in E(H)$ and list the correct values of $\rho'(a)$.

Case 1.1*:* $b_i = v = e_i$ for some pair $(b_i, e_i) \in \rho(a)$. This means that the signature$\rho(a)$ truncated to separation (A, B) must look exactly like $\rho(a)$, but without this subpath of length one. Thus we have one possible value for $\rho'(a)$, being $\rho(a)$ with this pair deleted.

Case 1.2*:* $b_i = v \neq e_i$ for some pair $(b_i, e_i) \in \rho(a)$. This means that the signature $\rho(a)$ truncated to separation (A, B) has to look the same as $\rho(a)$ but for the path corresponding to this very pair, which needs to be truncated by vertex v. The new beginning has to be either a vertex in $A \cap B$, or a forgotten vertex from $A \setminus B$. As T is semi-complete and (A, B) is a separation, there is an arc from v to every vertex of $A \setminus B$. Therefore, in $\rho'(a)$ the pair (b_i, e_i) has to be replaced with (b'_i, e_i), where $b'_i = \mathfrak{F}$ or b'_i is any vertex of $A \cap B$ such that there is an arc (v, b'_i). All the vertices $b'_i \neq \mathfrak{F}$ obtained in this manner have to be pairwise different. Moreover, we impose a condition that for all $a \in E(H)$ for which some pair of form (v, e_i) has been truncated to $(\mathfrak{F}, e_i)$, these pairs have to be pairwise non-equivalent with respect to $\equiv$ in ρ'; in this manner we forbid multiple usage of arcs going from v to the forgotten vertices.

Case 1.3*:* $b_i \neq v = e_i$ for some pair $(b_i, e_i) \in \rho(a)$. Similarly as before, signature $\rho(a)$ truncated to separation (A, B) has to look the same as $\rho(a)$ but for the path corresponding to this very pair, which needs to be truncated by vertex v. As (A, B) is a separation, the previous vertex on the path has to be in the separator $A \cap B$. Therefore, in $\mathcal{G}$ we take into consideration all signatures ρ' such that in ρ' the pair (b_i, e_i) is replaced with (b_i, e'_i), where e'_i is any vertex of $A \cap B$ such that there is an arc (e'_i, v). Also, all the vertices e'_i used in this manner for all $a \in E(H)$ have to be pairwise different.

Case 1.4*:* v is not contained in any pair in $\rho(a)$. Either v lies in the interior of some subpath from $\overline{\rho}(a)$, or it is not used by $\overline{\rho}(a)$ at all. In the first case the corresponding path in partial immersion on $(A \cup \{v\}, B)$ has to be split into two subpaths when truncating the immersion to (A, B). Along this path, the arc that was used to access v had to come from inside $A \cap B$, due to $(A \cup B)$ being a separation. However, the arc used to leave v can go to $A \cap B$ as well as to any forgotten vertex from $A \setminus B$, since (A, B) is a separation and T is a semi-complete digraph. In the second case, the signature of the truncated immersion stays the same. Therefore, in $\mathcal{G}$ we take into consideration signatures ρ' such that they not differ from ρ on a, or in $\rho'(a)$ exactly one pair (b_i, e_i) is replaced with two pairs (b_i, e'_i) and (b'_i, e_i), where $e'_i \in A \cap B$ with $(e'_i, v) \in E(T)$, whereas $b'_i = \mathfrak{F}$ or $b'_i \in A \cap B$ with $(v, b'_i) \in E(T)$. Similarly as before, all vertices $b'_i \neq \mathfrak{F}$ used have to be pairwise different and different from those used in Case 1.2, all vertices e'_i used have to be pairwise different and different from those used in Case 1.3, and all the pairs $(\mathfrak{F}, e_i)$ created in this manner have to be pairwise non-equivalent and non-equivalent to those created in Case 1.2 (with respect to $\equiv$ in ρ').

Case 2*:* $v = \rho(u)$ for some $u \in V(H)$. For every $(u, u') \in E(H)$, v has to be the beginning of the first pair of $\rho((u, u'))$; otherwise, $D_{(A \cup \{v\}, B)} = \bot$. Similarly, for every $(u', u) \in E(H)$, v

has to be the end of the last pair of $\rho((u', u))$; otherwise, $D_{(A\cup\{v\},B)}[\rho] = \bot$. Assuming both of these assertions hold, into $\mathcal{G}$ we can take all signatures ρ' such that: (i) ρ' differs on $V(H)$ from ρ only by having $\rho'(u) = \mathfrak{U}$; and (ii) the images from $\rho'(E(H))$ follow exactly the same rules as in Cases 1.1-1.4.

Forget vertex step Let us forget vertex $w \in A \setminus B$ from the separation $(A, B \cup \{w\})$, i.e., we consider the new separation (A, B). Let $\rho \in \mathcal{E}_{(A,B)}$; we argue that $D_{(A,B)}[\rho] = \bigvee_{\rho' \in \mathcal{G}} D_{(A\cup\{w\},B)}[\rho']$ for some set $\mathcal{G} \subseteq \mathcal{E}_{(A\cup\{w\},B)}$, which corresponds to possible extensions of ρ to the previous separation of larger order. Formally, one needs to argue that (i) if there exists a partial immersion with signature ρ on (A, B), then this partial immersion has signature $\rho' \in \mathcal{G}$ on $(A, B \cup \{w\})$, and (ii) if there exists a partial immersion with signature ρ' on $(A, B \cup \{w\})$ such that $\rho' \in \mathcal{G}$, then the signature of this partial immersion on (A, B) is ρ. Since this check in each case follows directly from the explanations provided along with the construction of $\mathcal{G}$, we leave the formal verification of this statement to the reader.

We now discuss which signatures ρ' are needed in $\mathcal{G}$ by considering all the signatures $\rho' \in \mathcal{E}_{(A\cup\{w\},B)}$ partitioned with respect to behaviour of the vertex w. For a fixed signature ρ' we put constraints on how ρ' must look like to be included in $\mathcal{G}$: we first put a constraint on the image of $V(H)$ in ρ', and then for every $a \in E(H)$ we list the possible values of $\rho'(a)$. In $\mathcal{G}$ we take into the account all signatures ρ' that (i) satisfy the imposed constraints on the image of $V(H)$, (ii) have one of the correct values for every $a \in E(H)$, and (iii) satisfy some additional, global constraints that are described later.

Case 1*:* $w \notin \rho'(V(H))$, that is, w is not an image of a vertex of H. In this case we require that $\rho'|_{V(H)} = \rho|_{V(H)}$. We now examine one arc $a \in E(H)$ and list the correct values of $\rho'(a)$ by looking at all possible values of $\rho'(a)$ partitioned with respect to behaviour on w.

Case 1.1*:* $b_i = w = e_i$ for some pair $(b_i, e_i) \in \rho'(a)$. This means that in the corresponding partial immersions w had to be left to $B \setminus A$; however, in T there is no arc from w to $B \setminus A$ as (A, B) is a separation. Therefore, in $\mathcal{G}$ we consider no signatures ρ' behaving in this manner on any arc a.

Case 1.2*:* $b_i = w \neq e_i$ for some pair $(b_i, e_i) \in \rho'(a)$. If we are to consider ρ' in $\mathcal{G}$, then w must prolong some path from the signature ρ in such a manner, that w is its beginning. After forgetting w the beginning of this path belongs to the forgotten vertices; therefore, in $\mathcal{G}$ we consider only signatures ρ' in which $\rho'(a)$ differs from $\rho(a)$ on exactly one pair: in $\rho'(a)$ there is (w, e_i) instead of $(\mathfrak{F}, e_i)$ in ρ. Moreover, all such pairs $(\mathfrak{F}, e_i)$ that were extended by w have to form one whole equivalence class with respect to $\equiv$ in ρ.

Case 1.3*:* $b_i \neq w = e_i$ for some pair $(b_i, e_i) \in \rho'(a)$. This means that in the corresponding partial expansions w have to be left to $B \setminus A$; however, in T there is no arc from w to $B \setminus A$ since (A, B) is a separation. We obtain a contradiction; therefore, in $\mathcal{G}$ we consider no signatures ρ' behaving in this manner on any arc a.

Case 1.4*:* w is not contained in any pair in $\rho'(a)$. In this case, in the corresponding partial expansions w has to be either unused by the trace of a, or be an internal vertex of a path in this trace. In both cases the signatures on $(A, B \cup \{w\})$ and on (A, B) are equal. It follows that in $\mathcal{G}$ we can consider only signatures with $\rho'(a) = \rho(a)$ for such arcs a.

Case 2*:* $w = \rho'(u)$ for some $u \in V(H)$. We consider in $\mathcal{G}$ signatures ρ' that differ from ρ in following manner: (i) ρ' differs on $V(H)$ from ρ only by having $\rho'(u) = w$ for exactly one vertex $u \in V(H)$ whereas $\rho(u) = \mathfrak{F}$; (ii) for all arcs $(u, u') \in E(H)$ the first pair of $\rho'((u, u'))$ is of form (w, e_1), whereas the first pair of $\rho((u, u'))$ is of form $(\mathfrak{F}, e_1)$; (iii) for all arcs $(u', u) \in E(H)$ the last pair of $\rho'((u', u))$ is of form (b_h, w), whereas the last pair of $\rho((u', u))$ is of form $(b_h, \mathfrak{F})$ (or $(\mathfrak{F}, \mathfrak{F})$ in case $b_h = w$); (iv) for all arcs $a \in E(H)$ non-incident with u we follow the same truncation rules as in Cases 1.1-1.4. Moreover, all the pairs in which w has been replaced with $\mathfrak{F}$ marker have to form one whole equivalence class with respect to $\equiv$.

Since updating the table $D_{(A,B)}$ for every separation (A, B) requires at most $\mathcal{O}(|\mathcal{E}_{(A,B)}|^2) = 2^{\mathcal{O}(k \log p + p\ell(\log \ell + \log p))}$ steps, while the number of separations in the pathwidth decomposition is $\mathcal{O}(n)$, the theorem follows. □

11 Conclusions and open problems

In this paper we have presented new methods of computing the two main width measures of semi-complete digraphs: cutwidth and pathwidth. For both of the width measures we have designed polynomial-time approximation and FPT exact algorithms.

We leave a number of open questions about the complexity of computing the cutwidth and the pathwidth of a semi-complete digraph. To begin with, we are still lacking proofs that computing the cutwidth or the pathwidth of a semi-complete digraph exactly is NP-hard. It is very hard to imagine that any of these problems could be solved in polynomial-time; however, proving NP-hardness of computing width measures is often technically very challenging. For example, NP-hardness of computing the cliquewidth of an undirected graph has been shown only in 2006 by Fellows et al. [24], after resisting attacks for a few years as a long-standing open problem. Furthermore, proving NP-hardness results for problems on tournaments is also known to be extremely difficult, because the instance obtained in the reduction is already very much constrained by the fact that it must be a tournament. The Feedback Arc Set in Tournaments problem, which is a simpler relative of the problem of computing exactly the cutwidth of a tournament, was proven to be NP-hard only recently. First, Ailon et al. proved that the problem is NP-hard under randomized reductions [1], and then Alon [2] and Charbit et al. [9] independently presented proofs that used only deterministic reductions.

When it comes to approximation algorithms, the obvious open question is to determine whether cutwidth admits a constant factor approximation; recall that the algorithm presented in this paper is only an $\mathcal{O}(OPT)$-approximation. As far as pathwidth is concerned, it is natural to ask if the parameter admits a PTAS, or whether computing it is APX-hard. Obviously, we need first to answer the question about NP-hardness, which is still unclear.

The exact algorithms also leave a room for improvement. For pathwidth it is natural to ask for an algorithm working in $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ time. Currently, the additional $\log k$ factor in the exponent is an artefact of using quadratic kernel of Buss for identifying sets of candidates for consecutive bags. If one was able for every bag to find a set of candidates for elements of this bag which was of size linear in k instead of quadratic, then an algorithm working in time $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ would immediately follow. As far as cutwidth is concerned, the $\log k$ factor under the square root in the exponent seems also artificial. At this moment, the appearance of this factor is a result of pipelining Lemma 5.13 with Lemma 5.16 in the proof of Lemma 5.17. A closer examination of the proofs of Lemmas 5.13 and 5.16 shows that bounds given by them are essentially optimal on their own; yet, it is not clear whether the bound given by pipelining them is optimal as well.

We have shown how our algorithms for pathwidth and cutwidth can be used to design FPT algorithms for testing various containment relations, and to obtain some exemplary meta-results.

We have also shown how to design an irrelevant vertex rule on a triple for rerouting a family of arc-disjoint paths, which leads to an FPT algorithm for the ROOTED IMMERSION problem. It is noteworthy that the presented algorithms have much better running time guarantees than their famous analogues in the Graph Minors series of Robertson and Seymour. For instance, the celebrated algorithm testing whether H is a minor of G [47] runs in time $f(|H|) \cdot |V(G)|^3$ for a gargantuan function f that is not even specified; in particular, f is not elementary. The algorithms presented in this paper have good dependency both on the size of the digraph to be embedded (single exponential), and on the size of the semi-complete digraph into which we embed (linear). This is mostly thanks to the usage of the new set of obstacles, especially the short jungles.

The natural question here is whether this new set of obstacles, and in particular the short jungles, can give raise to more powerful irrelevant vertex rules. For example, if we consider the ROOTED IMMERSION problem, it is tempting to try to replace finding an irrelevant vertex in a triple with a direct irrelevant vertex rule on a short jungle of size polynomial in the size of the digraph to be immersed. If this was possible, the running time of the algorithm for ROOTED IMMERSION could be reduced to single-exponential in terms of the size of the tested pattern digraph.

Perhaps a much more important question is whether one can also design an FPT algorithm for the rooted variant of topological containment testing, called further ROOTED TOPOLOGICAL CONTAINMENT, since this was possible for ROOTED IMMERSION. Observe that the classical VERTEX DISJOINT PATHS problem is in fact a special case of ROOTED TOPOLOGICAL CONTAINMENT where all the vertices of H have specified images. Chudnovsky, Scott and Seymour [12] proved that k-VERTEX DISJOINT PATHS is solvable in time $n^{\mathcal{O}(f(k))}$, for some function f, i.e is in class XP, on semi-complete digraphs. Their results also imply that the ROOTED TOPOLOGICAL CONTAINMENT is in XP on semi-complete digraphs. To the best of our knowledge, the question whether VERTEX DISJOINT PATHS on semi-complete digraphs can be solved in FPT time is still open.

Unfortunately, our approach used for ROOTED IMMERSION does not apply directly to this problem. Dynamic programming on path decomposition works fine but the problem is with the irrelevant vertex arguments. Even for $k = 2$ there exist tournaments that contain arbitrarily large triples, but in which every vertex is relevant; let us now provide such an example.

For even n we construct a tournament T_n with two pairs of vertices (s_1, t_1), (s_2, t_2) so that the following properties are satisfied:

(i) T_n contains an $(n/2 - 1)$-triple;

(ii) there is exactly one solution to VERTEX DISJOINT PATHS instance $(T_n, \{(s_1, t_1), (s_2, t_2)\})$ in which all the vertices of $V(T_n)$ lie on one of the paths.

This example shows that even though a graph can be complicated from the point of view of path decompositions, all the vertices can be relevant.

Let $V(T_n) = \{a_i, b_i : 1 \leq i \leq n\}$, where $s_1 = a_1$, $t_1 = a_n$, $s_2 = b_n$ and $t_2 = b_1$. Construct following arcs:

- for every $i \in \{1, 2, \ldots, n-1\}$ create an arc (a_i, a_{i+1});
- for every $i, j \in \{1, 2, \ldots, n\}, i < j - 1$ create an arc (a_j, a_i);
- for every $i \in \{1, 2, \ldots, n-1\}$ create an arc (b_{i+1}, b_i);
- for every $i, j \in \{1, 2, \ldots, n\}, i > j + 1$ create an arc (b_j, b_i);

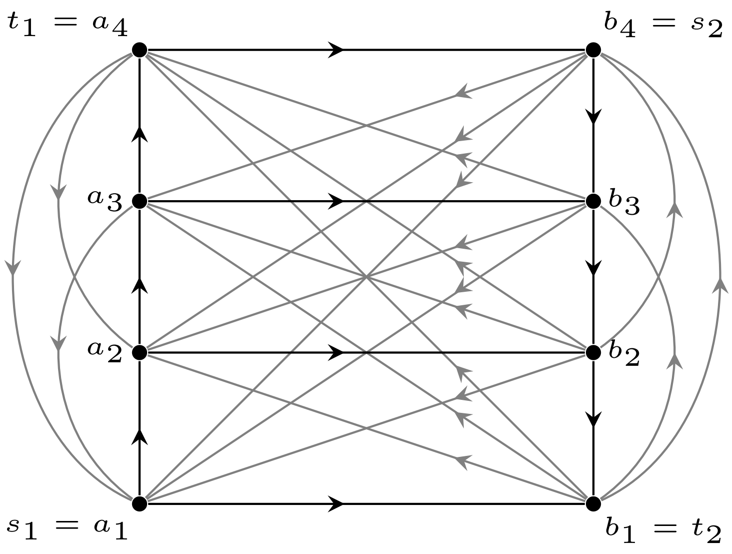

Figure 5: Tournament T_4.

- for every $i \in \{1, 2, \ldots, n\}$ create an arc (a_i, b_i);
- for every $i, j \in \{1, 2, \ldots, n\}, i \neq j$ create an arc (b_j, a_i).

To see that T_n satisfies (i), observe that

$$\begin{array}{rl}(& \{b_1, b_2, \ldots, b_{n/2-1}\}, \\ & \{a_{n/2+1}, a_{n/2+2}, \ldots, a_n\}, \\ & \{a_1, a_2, \ldots, a_{n/2-1}\} \quad)\end{array}$$

is a $(n/2 - 1)$-triple. To prove that (ii) is satisfied as well, observe that there is a solution to Vertex Disjoint Paths problem containing two paths $(a_1, a_2, \ldots, a_n)$ and $(b_n, b_{n-1}, \ldots, b_1)$ which in total use all the vertices of T_n. Assume that there is some other solution and let k be the largest index such that the path connecting a_1 to a_n begins with prefix $(a_1, a_2, \ldots, a_k)$. As the solution is different from the aforementioned one, it follows that $k < n$. Therefore, the next vertex on the path has to be b_k, as this is the only unused outneighbor of a_k apart from a_{k+1}. But if the path from a_1 to a_n already uses $\{a_1, a_2, \ldots, a_k, b_k\}$, we see that there is no arc going from $\{a_{k+1}, a_{k+2}, \ldots, a_n, b_{k+1}, b_{k+2}, \ldots, b_n\}$ to $\{b_1, b_2, \ldots, b_{k-1}\}$, so we are already unable to construct a path from b_n to b_1. This is a contradiction.

The presented example suggests that a possible way of obtaining an FPT algorithm for Vertex Disjoint Paths problem requires another width parameter admitting more powerful obstacles.

Acknowledgement. The authors thank Stéphan Thomassé for an excellent remark after the presentation during STACS 2013, which enabled to prove the bound $4k+2$ in Lemma 3.8 instead of original $5k + 2$. Consequence of this better bound is approximation ratio 6 for pathwidth of a semi-complete digraph, instead of original 7.

References

[1] N. Ailon, M. Charikar, and A. Newman, *Aggregating inconsistent information: Ranking and clustering*, J. ACM, 55 (2008).

[2] N. Alon, *Ranking tournaments*, SIAM J. Discrete Math., 20 (2006), pp. 137–142.

[3] N. Alon, D. Lokshtanov, and S. Saurabh, *Fast FAST*, in ICALP (1), vol. 5555 of Lecture Notes in Computer Science, Springer, 2009, pp. 49–58.

[4] J. Bang-Jensen, *Edge-disjoint in- and out-branchings in tournaments and related path problems*, J. Comb. Theory, Ser. B, 51 (1991), pp. 1–23.

[5] J. Bang-Jensen and G. Gutin, *Digraphs - theory, algorithms and applications*, Springer Monographs in Mathematics, Springer-Verlag London Ltd., London, second ed., 2009.

[6] J. Bang-Jensen and C. Thomassen, *A polynomial algorithm for the 2-path problem for semicomplete digraphs*, SIAM J. Discrete Math., 5 (1992), pp. 366–376.

[7] D. Berwanger, A. Dawar, P. Hunter, S. Kreutzer, and J. Obdržálek, *The DAG-width of directed graphs*, J. Comb. Theory, Ser. B, 102 (2012), pp. 900–923.

[8] J. F. Buss and J. Goldsmith, *Nondeterminism within P*, SIAM J. Comput., 22 (1993), pp. 560–572.

[9] P. Charbit, S. Thomassé, and A. Yeo, *The minimum feedback arc set problem is NP-hard for tournaments*, Combinatorics, Probability & Computing, 16 (2007), pp. 1–4.

[10] P. Z. Chinn, J. Chvátalová, A. K. Dewdney, and N. E. Gibbs, *The bandwidth problem for graphs and matrices — a survey*, J. Graph Theory, 6 (1982), pp. 223–254.

[11] M. Chudnovsky, A. Fradkin, and P. D. Seymour, *Tournament immersion and cutwidth*, J. Comb. Theory Ser. B, 102 (2012), pp. 93–101.

[12] M. Chudnovsky, A. Scott, and P. D. Seymour, *Disjoint paths in tournaments*, 2010. Manuscript.

[13] M. Chudnovsky and P. D. Seymour, *A well-quasi-order for tournaments*, J. Comb. Theory, Ser. B, 101 (2011), pp. 47–53.

[14] B. Courcelle and J. Engelfriet, *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*, vol. 138 of Encyclopedia of mathematics and its applications, Cambridge University Press, 2012.

[15] B. Courcelle, J. A. Makowsky, and U. Rotics, *Linear time solvable optimization problems on graphs of bounded clique-width*, Theory Comput. Syst., 33 (2000), pp. 125–150.

[16] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos, *Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs*, J. ACM, 52 (2005), pp. 866–893.

[17] J. Díaz, J. Petit, and M. J. Serna, *A survey of graph layout problems*, ACM Comput. Surv., 34 (2002), pp. 313–356.

[18] R. Diestel, *Graph Theory, 4th Edition*, vol. 173 of Graduate texts in mathematics, Springer, 2012.

[19] R. G. Downey and M. R. Fellows, *Parameterized complexity*, Springer-Verlag, New York, 1999.

[20] P. Erdős, *On an elementary proof of some asymptotic formulas in the theory of partitions*, Annals of Mathematics (2), 43 (1942), pp. 437–450.

[21] P. Erdős and G. Szekeres, *A combinatorial problem in geometry*, Compositio Math., 2 (1935), pp. 463–470.

[22] S. Even, A. Itai, and A. Shamir, *On the complexity of timetable and multicommodity flow problems*, SIAM J. Comput., 5 (1976), pp. 691–703.

[23] U. Feige, *Faster FAST (Feedback Arc Set in Tournaments)*, CoRR, abs/0911.5094 (2009).

[24] M. R. Fellows, F. A. Rosamond, U. Rotics, and S. Szeider, *Clique-width is NP-complete*, SIAM J. Discrete Math., 23 (2009), pp. 909–939.

[25] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin, 2006.

[26] F. V. Fomin, S. Kratsch, M. Pilipczuk, M. Pilipczuk, and Y. Villanger, *Tight bounds for parameterized complexity of Cluster Editing*, in STACS 2013, vol. 20 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013, pp. 32–43.

[27] F. V. Fomin and M. Pilipczuk, *Jungles, bundles, and fixed-parameter tractability*, in SODA 2013, SIAM, 2013, pp. 396–413.

[28] ——, *Subexponential parameterized algorithm for computing the cutwidth of a semi-complete digraph*, in ESA 2013, vol. 8125 of Lecture Notes in Computer Science, Springer, 2013, pp. 505–516.

[29] S. Fortune, J. E. Hopcroft, and J. Wyllie, *The directed subgraph homeomorphism problem*, Theoretical Computer Science, 10 (1980), pp. 111–121.

[30] A. Fradkin and P. D. Seymour, *Edge-disjoint paths in digraphs with bounded independence number*, 2010. Manuscript.

[31] A. O. Fradkin and P. D. Seymour, *Tournament pathwidth and topological containment*, J. Comb. Theory, Ser. B, 103 (2013), pp. 374–384.

[32] R. Ganian and P. Hliněný, *On parse trees and Myhill-Nerode-type tools for handling graphs of bounded rank-width*, Discrete Applied Mathematics, 158 (2010), pp. 851–867.

[33] R. Ganian, P. Hliněný, J. Kneis, D. Meister, J. Obdržálek, P. Rossmanith, and S. Sikdar, *Are there any good digraph width measures?*, in IPEC 2010, vol. 6478 of Lecture Notes in Computer Science, Springer, 2010, pp. 135–146.

[34] R. Ganian, P. Hliněný, and J. Obdržálek, *Clique-width: When hard does not mean impossible*, in STACS 2011, vol. 9 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011, pp. 404–415.

[35] M. Grohe, K. Kawarabayashi, D. Marx, and P. Wollan, *Finding topological subgraphs is fixed-parameter tractable*, in STOC 2011, ACM, 2011, pp. 479–488.

[36] G. H. Hardy and S. Ramanujan, *Asymptotic formulae in combinatory analysis*, Proceedings of the London Mathematical Society, s2-17 (1918), pp. 75–115.

[37] P. Hunter and S. Kreutzer, *Digraph measures: Kelly decompositions, games, and orderings*, Theoretical Computer Science, 399 (2008), pp. 206–219.

[38] T. Johnson, N. Robertson, P. D. Seymour, and R. Thomas, *Directed tree-width*, J. Combin. Theory Ser. B, 82 (2001), pp. 138–154.

[39] M. M. Kanté and M. Rao, *The rank-width of edge-coloured graphs*, Theory Comput. Syst., 52 (2013), pp. 599–644.

[40] M. Karpinski and W. Schudy, *Faster algorithms for Feedback Arc Set Tournament, Kemeny Rank Aggregation and Betweenness Tournament*, in ISAAC 2010, vol. 6506 of Lecture Notes in Computer Science, Springer, 2010, pp. 3–14.

[41] D. Kőnig, *Gráfok és mátrixok*, Matematikai és Fizikai Lapok, 38 (1931), pp. 116–119. In Hungarian.

[42] I. Kim and P. D. Seymour, *Tournament minors*, CoRR, abs/1206.3135 (2012).

[43] M. Pilipczuk, *Computing cutwidth and pathwidth of semi-complete digraphs via degree orderings*, in STACS 2013, vol. 20 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013, pp. 197–208.

[44] ——, *Tournaments and Optimality: New Results in Parameterized Complexity*, PhD thesis, University of Bergen, Norway, 2013.

[45] M. D. Plummer and L. Lovász, *Matching Theory*, ACM Chelsea Publishing, Providence, Rhode Island, 2009.

[46] V. Raman and S. Saurabh, *Parameterized algorithms for feedback set problems and their duals in tournaments*, Theoretical Computer Science, 351 (2006), pp. 446–458.

[47] N. Robertson and P. D. Seymour, *Graph Minors. XIII. The Disjoint Paths problem*, J. Comb. Theory, Ser. B, 63 (1995), pp. 65–110.

[48] ——, *Graph Minors. XX. Wagner's conjecture*, J. Comb. Theory, Ser. B, 92 (2004), pp. 325–357.

[49] A. Slivkins, *Parameterized tractability of edge-disjoint paths on directed acyclic graphs*, SIAM J. Discrete Math., 24 (2010), pp. 146–157.

[50] D. M. Thilikos, M. J. Serna, and H. L. Bodlaender, *Cutwidth I: A linear time fixed parameter algorithm*, J. Algorithms, 56 (2005), pp. 1–24.