UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS
DEPARTAMENTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RAUL WAYNE TEIXEIRA LOPES

DISJOINT PATHS AND THE GRID THEOREM IN DIGRAPHS

FORTALEZA
2021

RAUL WAYNE TEIXEIRA LOPES

DISJOINT PATHS AND THE GRID THEOREM IN DIGRAPHS

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Algoritmos.

Orientador: Prof. Dr. Victor Almeida Campos

FORTALEZA

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

L855d Lopes, Raul Wayne Teixeira.
Disjoint paths and the Grid Theorem in Digraphs / Raul Wayne Teixeira Lopes. – 2021.
99 f. : il.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação , Fortaleza, 2021.
Orientação: Prof. Dr. Victor Almeida Campos.

1. Complexidade parameterizada. 2. Digrafos. 3. Teorema do Grid. 4. Largura em árvore direcionada. 5. Kernelização. I. Título.

CDD 005

RAUL WAYNE TEIXEIRA LOPES

DISJOINT PATHS AND THE GRID THEOREM IN DIGRAPHS

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Algoritmos.

Aprovada em: 23/06/2021

ACKNOWLEDGEMENTS

To my family and my friends, for the support, motivation, and gaming sessions. In special, to my grandmother Ziza for being an amazing example and for, well, being Ziza. To my late grandfather José Colombo, also for being an amazing example. You were the best man I have ever known. And to my wife Natália Diógenes, who always believed in me, supported me, and pushed me forward.

To everyone I had the pleasure of working and learning with so far: Alexsander Melo, Allen Ibiapina, Ana Karolinna Maia de Oliveira, Ana Shirley, Andrea Marino, Celina Figueiredo, Cláudia Linhares, Cláudio Carvalho, Ignasi Sau, Jonas Costa, Guilherme Gomes, Nicolas Nisse, Victor Campos, and others.

To the friends I made in Montpellier, for the working environment, the friendship, and all those boardgames matches: Alan Carneiro, Kyllia de Paiva, Guilherme Gomes, and Jacqueline Torres.

To my supervisor, Victor Almeida Campos, for accepting me as student since I was an undergrad and for the many years of collaboration. A long time ago you said "he'll get there if he keeps going like this", and now here we are. To Ignasi Sau, for the collaboration and for teaching me, like, a lot, during the year I spent in Montpellier. You pushed me and supported me to climb a mountain. Literally. And to Frédéric Havet and Christophe Paul, for the patience and not leaving me behind in that mountain.

To all the members of the ParGO group, for the excellent study and research environment.

To the members of the evaluation committe, Ana Karolinna Maia de Oliveira, Rudini Menezes Sampaio, Uéverton dos Santos Souza, Vinícius Fernandes dos Santos, and Ignasi Sau, for helpful commentaries about the text, corrections, and suggestions.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

***

À minha família e amigos, pelo apoio, motivação e jogatinas. Em especial, à minha avó Ziza por ser um grande exemplo e por, bem, ser a Ziza. Ao meu avô José Colombo, por ser outro grande exemplo. Você foi o melhor homem que já conheci. E à minha esposa Natália Diógenes, que sempre acreditou em mim, me apoiou e me incentivou.

A todos com quem tive o prazer de trabalhar e aprender até agora: Alexsander Melo, Allen Ibiapina, Ana Karolinna Maia de Oliveira, Ana Shirley, Andrea Marino, Celina Figueiredo, Cláudia Linhares, Cláudio Carvalho, Ignasi Sau, Jonas Costa, Guilherme Gomes, Nicolas Nisse, Victor Campos e outros.

Aos amigos que fiz em Montpellier, pelo ambiente de trabalho, pela amizade e por todos aquelas partidas de boardgames: Alan Carneiro, Kyllia de Paiva, Guilherme Gomes e Jacqueline Torres.

Ao meu orientador, Victor Almeida Campos, por me aceitar como aluno desde a graduação e pelos muitos anos de colaboração. Há muito tempo você disse "ele vai chegar lá se continuar assim", e agora aqui estamos. Ao Ignasi Sau, pela colaboração e por me ensinar, tipo, muita coisa, durante o ano que passei em Montpellier. Você me empurrou e me incentivou a cruzar uma montanha. Literalmente. E a Frédéric Havet e Christophe Paul, pela paciência e por não me abandonarem naquela montanha.

A todos os membros do grupo ParGO, pelo excelente ambiente de estudo e pesquisa.

Aos membros da comissão avaliadora, Ana Karolinna Maia de Oliveira, Rudini Menezes Sampaio, Uéverton dos Santos Souza, Vinícius Fernandes dos Santos e Ignasi Sau, pelos comentários sobre o texto, correções e sugestões.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

RESUMO

Em complexidade parametrizada, frequentemente nos deparamos com problemas FPT em grafos não direcionados que se tornam W[1]-difíceis quando adaptados para o caso direcionado. O problema de Caminhos Direcionados e Disjuntos, um problema notoriamente difícil em digrafos, é um clássico exemplo disto: Robertson e Seymour [1] mostraram que a versão não direcionada deste problema é FPT quando parametrizada pelo número k de demandas mas, por outro lado, Fortune et al. [2] mostraram que a versão direcionada é NP-completa para $k = 2$ fixo e Slivkins [3] mostrou que é W[1]-difícil com relação ao parâmetro k mesmo quando restrito a digrafos acíclicos. Nesta tese, nós mostramos dois novos resultados relacionados a complexidade parametrizada em digrafos. Primeiro, mostramos como adaptar o Teorema do Grid Cilíndrico de Kawarabayashi e Kreutzer [4], um resultado análogo ao celebrado Teorema do Grid de Robertson e Seymour [5], em um algoritmo FPT. Depois, nós introduzimos uma nova relaxação de Caminhos Direcionados e Disjuntos onde pedimos apenas que os caminhos comportem-se adequadamente não no digrafo inteiro, mas em uma parte não especificada de tamanho dado por um parâmetro. Isto é, no problema de Caminhos Direcionados Suficientemente Disjuntos, recebemos um digrafo D junto com um conjunto de k pares de vértices (as demandas) e dois inteiros não-negativos k e s, e o objetivo é encontrar uma coleção de caminhos ligando as demandas tal que pelo menos d vértices de D ocorram em no máximo s caminhos da coleção. Entre outros resultados algorítmicos e de dificuldade, mostramos que este problema tem um kernel de tamanho $d \cdot 2^{k-s} \cdot \binom{k}{s} + 2k$ em digrafos. Este resultado tem consequências para o problema de Rede de Steiner: nós mostramos que este é FPT parametrizado pelo número k de terminais e p, onde $p = n - q$ e q é o tamanho da solução.

Palavras-chave: complexidade parametrizada; digrafos; Teorema do Grid; largura em árvore direcionada; kernelização.

ABSTRACT

In parameterized complexity, it is often the case that FPT problems in undirected graphs become W[1]-hard when translated to the directed setting. The DIRECTED DISJOINT PATHS problem, a notoriously hard problem in digraphs, is a classical example of this: Robertson and Seymour [1] showed that undirected version is FPT when parameterized by the number k of requests but, on the other hand, Fortune et al. [2], showed that the directed version is NP-complete for fixed $k = 2$ and Slivkins [3] showed that it is W[1]-hard with parameter k even when restricted to acyclic digraphs. In this thesis, we provide two new results regarding parameterized complexity in digraphs. First, we show how to adapt the Directed Grid Theorem by Kawarabayashi and Kreutzer [4], a result analogous to the celebrated Grid Theorem by Robertson and Seymour [5], into an FPT algorithm. Then, we introduce a novel relaxation for DIRECTED DISJOINT PATHS in which we only require the paths to behave properly not in the whole digraph, but in an unspecified part of size prescribed by a parameter. Namely, in the DISJOINT ENOUGH DIRECTED PATHS problem, given a digraph D together with a set of k pairs of vertices (the requests) and two non-negative integers k and s, the task is to find a collection of paths connecting the requests such that at least d vertices of D occur in at most s paths of the collection. Amongst other algorithmic and hardness we show that this problem has a kernel of size $d \cdot 2^{k-s} \cdot \binom{k}{s} + 2k$ in general digraphs. This result has consequences for the STEINER NETWORK problem: we show that it is FPT parameterized by the number k of terminals and p, where $p = n - q$ and q is the size of the solution.

Keywords: parameterized complexity; digraphs; Grid Theorem; directed tree-width; kernelization.

LIST OF FIGURES

LIST OF SYMBOLS

$D(\mathcal{P})$	Digraph formed by the union of all paths in $\mathcal{P}$.
$d_D^-(v)$	In-degree of v in the digraph D
$d_D^+(v)$	Out-degree of v in the digraph D
$d_D(v)$	Degree of v in the digraph D
$\mathsf{dtw}(D)$	Directed tree-width of D
$G[X]$	Graph induced by X
$G \setminus X$	Graph resulting by deleting X from G
$H \subseteq G$	Subgraph relation
$[\ell]$	Set $\{1, \dots, \ell\}$
$N_D^-(v)$	In-neighborhood of v in D
$N_D^+(v)$	Out-neighborhood of v in D
n	Number of vertices of the input (di)graph
m	Number of edges of the input (di)graph
(u,v)	Edge from u to v
$\mathsf{Stirling}(a,b)$	Stirling number of the second kind $(\frac{1}{2}\binom{a}{b} \cdot b^{a-b})$

SUMMARY

1 OVERVIEW

Parameterized complexity offers an approach to deal with computational problems that are unlikely to admit polynomial time algorithms. Here, we express the running time of an algorithm for a given problem as a function depending on the size of the input and on a *parameter* carrying some information regarding instances of the problem, such as the size of the solution, that can be relatively small with respect to the size of the input.

For a decision problem Π with input size n and an associated parameter k, the goal is to provide an algorithm $\mathcal{A}$ with running time of the form $f(k) \cdot n^{\mathcal{O}(1)}$ for a computable function f. In this case, we say that Π is *fixed-parameter tractable* (FPT) with respect to parameter k, and that $\mathcal{A}$ is an *FPT algorithm.* If an algorithm $\mathcal{A}$ with running time $f(k) \cdot n^{g(k)}$ exists for computable functions f and g, we say that Π is *slice-wise polynomial* (XP) with parameter k, and that $\mathcal{A}$ is an *XP algorithm.*

Notice that XP and FPT algorithms run in polynomial time for *fixed* k. In the particular case of FPT algorithms, the idea is to isolate the combinatorial explosion that is likely to occur when solving NP-hard problems by a function depending on the parameter of analysis only, and there is a big difference in the running times of FPT and XP algorithms in favor of the formers. To further the discussion on parameterized complexity, we now briefly introduce two classical NP-complete problems.

In the VERTEX COVER problem, we are given an n-vertex graph G together with a non-negative integer k, and the task is to decide if there is a set of vertices X with $|X| \leq k$ such that every edge of G has an endpoint in X. This problem admits a simple $2^k \cdot n^{\mathcal{O}(1)}$ algorithm [6, Chapter 3] and therefore it is FPT. The situation is different, however, for the CLIQUE[1] problem, in which we receive as input a graph G together with a positive integer k and the task is to decide if G has a clique of size at least k. Although a classical result by Karp [7] states that CLIQUE is NP-complete, it can be solved in $\mathcal{O}(n^k)$ time by checking if any of the $\binom{|V(G)|}{k}$ subsets of $V(G)$ with size k is a clique. Thus, there is an $\mathcal{O}(k^2 \cdot n^k)$ algorithm for CLIQUE and it is XP with respect to parameter k. On the other hand, there is strong evidence that CLIQUE is *not* FPT regarding this parameterization.

Within parameterized complexity, the W-hierarchy can be seen as the parameterized equivalent of the class NP of classical decision problems, a problem being W[1]-hard can be

[1] A *clique* in a graph $G = (V, E)$ is a set $X \subseteq V(G)$ such that there is an edge in $E(G)$ between any pair of distinct vertices in X.

seen as a strong evidence that it is not FPT, and the CLIQUE problem with parameter k is the canonical example of a W[1]-hard problem. Downey and Fellows [8] showed that CLIQUE and INDEPENDENT SET are both W[1]-complete when parameterized by the size of the solution.

Additionally, some problems are even more resistant to some parameterizations than W[1]-hard problems. In the COLORING problem, for example, we receive as input an undirected graph G together with a positive integer k and the goal is to color the vertices of G using at most k colors in such way that any pair of adjacent vertices receive different colors. A well-known result by Karp [7] states that COLORING is NP-complete for every fixed $k \geq 3$.

It is worth mentioning that some deterioration in the tractability of parameterized problems is often observed when moving from the undirected to the directed case. For example, Dreyfus and Wagner [9] showed that STEINER TREE is FPT when parameterized by the number of terminal vertices, and Bousquet et al. [10] and Marx and Razgon [11] showed that MULTICUT is FPT when parameterized by the size of the solution. On the other hand, Guo et al. [12] showed that a particular case of STEINER NETWORK, the directed counterpart of STEINER TREE, is W[1]-hard when parameterized by the number of terminal vertices, and Pilipczuk and Wahlström [13] showed that DIRECTED MULTICUT is W[1]-hard when parameterized by the size of the solution. Later, Feldmann and Marx [14] completely characterized the tractability of STEINER NETWORK with the aforementioned parameterization.

Finally, many hard problems become tractable under parameterizations that, although not directly linked to the problem, provide some structural properties on the input graph. For example, a number of hard problems can be efficiently solved in graphs of bounded *tree-width*, a parameter introduced by Bertele and Brioschi [15], then by Halin [16], and finally by Robertson and Seymour [5], that measures how tightly an undirected graph can be approximated by a tree.

In this thesis we present two new positive results regarding parameterized complexity in digraphs.

Adapting the Directed Grid Theorem into an FPT algorithm [17]. The celebrated Grid Theorem by Robertson and Seymour [5] states that there is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that every undirected graph with tree-width at least $f(k)$ contains a $(k \times k)$-grid as a minor. A polynomial bound for $f(k)$ was given by Chekuri and Chuzhoy [18] and later improved by Chuzhoy and Tan [19]. This has result found numerous applications in the design of FPT algorithms for problems in undirected graphs, amongst other usages (we refer the reader to [6, Chapter 7.7] for examples of applications).

In particular, the Grid Theorem allows the design of "win/win" approaches to attack parameterized problems in graphs where we either gain because we conclude that the input graph has bounded tree-width, or we gain by finding a large grid minor which is then used, for example, as a certificate that the instance is positive (or negative), or to find a vertex that can be safely deleted from the graph without changing the answer to the problem (thus reducing the size of the input). The first scenario is an example of application of the framework of *Bidimensionality* [20], introduced by Demaine et al. [21], a powerful tool that is often used to design FPT algorithms with subexponential dependence on the considered parameter in planar graphs. The second scenario is a rough description of the *irrevelant vertex* technique, originally introduced by Robertson and Seymour [1,22,23] to solve the DISJOINT PATHS problem.

Kawarabayashi and Kreutzer [4][2] proved a result that is analogous to the Grid Theorem for digraphs, exposing the duality between *directed tree-width*[3], a parameter introduced by Johnson et al. [24] which, informally, measures the distance of a digraph to being acyclic, and the existence of a large planar minor of a particular kind in digraphs. Namely, in [4] it is shown that every digraph with sufficiently large directed tree-width contains a large *cylindrical grid*[3] as a *butterfly minor*[3], and there is hope that this result has laid the foundation for the development of a framework analogous to bidimensionality for digraphs. In the first part of this thesis, we show how to adapt the Directed Grid Theorem into an FPT algorithm by making some local changes in the proof by Kawarabayashi and Kreutzer [4].

This is a joint work with Victor Campos, Ana Karolinna Maia, and Ignasi Sau. An extended abstract of this work is available in the *Proceedings of the X Latin and American Algorithms, Graphs, and Optimization Symposium (**LAGOS**), volume 346 of ENTCS, pages 229-240, 2019*, and the full version is currently submitted to *SIAM Journal on Discrete Mathematics* and awaiting review.

A relaxation of the Directed Disjoint Paths problem: a global congestion metric helps [25]. In the DISJOINT PATHS problem, we are given a graph G and a set of pairs of vertices $\{(s_1,t_1),\ldots,(s_k,t_k)\}$, called the *requests*, and the task is to find a collection of pairwise vertex-disjoint paths $\{P_1,\ldots,P_k\}$ such that each P_i is a path from s_i to t_i. The DIRECTED DISJOINT PATHS (DDP) problem is defined analogously, but in digraphs and we ask the paths to be directed. Although both undirected and directed versions are NP-complete even in planar

[2] The full version of [4] is available at CoRR abs/1411.5681.
[3] See Section 2.3 for the definitions of directed tree-width, cylindrical grids, and butterfly minors.

(di)graphs [2,26], the latter turns out to be significantly harder under the optics of parameterized complexity.

Robertson and Seymour [1] showed that DISJOINT PATHS is FPT with relation to parameter k but, on the other hand, Fortune et al. [2] showed that DIRECTED DISJOINT PATHS is NP-complete even for fixed $k = 2$. On the positive side, the authors also showed that this problem is XP with parameter k in acyclic digraphs (DAGs). Finally, Slivkins [3] showed that this parameterization is W[1]-hard even when restricted to DAGs and, by a simple local reduction shown by Amiri et al. [27], it is easy to extend Slivkins' reduction [3] to show that DIRECTED DISJOINT PATHS WITH CONGESTION c, where we allow each vertex of the digraph to appear in at most c paths of the collection, is also W[1]-hard with the same parameterization for every fixed $c \geq 1$.

Positive results for DIRECTED DISJOINT PATHS, however, are scarce and usually consider a restriction on the input digraph. For example, Johnson et al. [24] showed that DDP with parameter k is XP in digraphs of bounded directed tree-width, Schrijver [28] showed that it is also XP in planar digraphs, and then Cygan et al. [29] improved on this result by providing an FPT algorithm for this problem in planar digraphs.

In the second part of this thesis, we introduce a novel relaxation of the DIRECTED DISJOINT PATHS problem in which we consider a *global* congestion metric on top of a *local* one. We introduce the DISJOINT ENOUGH DIRECTED PATHS problem (DEDP) where, given as input a set of requests in a digraph D together with two non-negative integers d and s, we are asked to find a collection of paths linking the requests such that at least d vertices of D occur in at most s paths of the collection. Our main contribution regarding this problem is a kernel of size $d \cdot 2^{k-s} \cdot \binom{k}{s}$. See Table 1 for a summary of algorithmic and hardness results we prove for DEDP.

k	d	s	w	Complexity
fixed ≥ 3	$\Omega(n^{\alpha})$	fixed ≥ 1	—	NP-complete (Theorem 4.2.1)
parameter	$\Omega(n^{\alpha})$	fixed ≥ 1	0	W[1]-hard (Theorem 4.2.1)
input	parameter	fixed ≥ 0	—	W[1]-hard (Theorem 4.2.2)
parameter	—	—	parameter	XP (Theorem 4.3.10)
input	parameter	parameter	—	XP (Theorem 4.3.12)
parameter	parameter	parameter	—	FPT (Theorem 4.3.21)

Table 1 – Summary of hardness and algorithmic results for distinct choices of the parameters. A horizontal line in a cell means no restrictions for that case.

This is a joint work with Ignasi Sau. An extended abstract of this work is available in the *Proceedings of the 45th International Symposium on Mathematical Foundations of Computer Science (**MFCS**), volume 170 of LIPIcs, pages 68:1-68:15, 2020*, and the full version is to appear in *Theoretical Computer Science (**TCS**)*.

Organization. This thesis is organized as follows. In Chapter 2 we introduce the basic definitions and notations commonly used by all parts of this text, as well as some preliminary known results. In Chapter 3 we motivate, prove and further discuss our adaptation of the Directed Grid Theorem [4] into an FPT algorithm. In Chapter 4, we formally introduce the DISJOINT ENOUGH DIRECTED PATHS problem, prove our algorithmic and hardness results, discuss some consequences of these results, and pose some open problems. In Appendix A we include a list of all definitions used in this thesis and in Appendix B we include one page abstracts of other results obtained during the Ph.D.

2 DEFINITIONS AND PRELIMINARIES

We refer the reader to [30] for basic background on graph theory. A *graph G* is formed by an ordered pair $(V(G),E(G))$ where $V(G)$ is a set of *vertices*, $E(G)$ is a set of *edges*, and an *incidence function* $\psi_G : E(G) \to V(G) \times V(G)$ associating each edge $e \in E(G)$ with an unordered pair of vertices (u,v) of G. We say that $|V(G)|$ is the *size* of G. If there is an edge e with $\psi_G(e) = (u,v)$, we say that e is an edge *between u and v* and simply write $e = (u,v)$. We also say that u and v are *adjacent*, that e is *incident* to u and v, and refer to u and v as the *endpoints* of e. A *digraph D* is defined similarly, but with the difference that the incidence function ψ_D associates each edge $e \in E(D)$ with an *ordered* pair of vertices (u,v) of D. All the definitions for graphs given above also apply to digraphs. Additionaly, if e is an edge of D with $e = (u,v)$, we say that u is the *tail* of e, that v is the *head* of e, and that e is *oriented* from u to v. The *underlying graph* of D is the graph obtained by ignoring the orientation of the edges of D.

Unless stated otherwise, we let $n = |V(G)|$ and $m = |E(G)|$ when G is the input (di)graph of some procedure. For an integer $\ell \geq 1$, we denote by $[\ell]$ the set $\{1,2,\ldots,\ell\}$.

The *in-degree* $\deg_D^-(v)$ (resp. *out-degree* $\deg_D^+(v)$) of a vertex v in a digraph D is the number of edges with head (resp. tail) v. The *degree* $\deg_D(v)$ of v in D is the sum of $\deg_D^-(v)$ with $\deg_D^+(v)$. The *in-neighborhood* $N_D^-(v)$ of v is the set $\{u \in V(D) \mid (u,v) \in E(G)\}$, and the *out-neighborhood* $N_D^+(v)$ is the set $\{u \in V(D) \mid (v,u) \in E(G)\}$. We say that u is an *in-neighbor* of v if $u \in N_D^-(v)$ and that u is an *out-neighbor* of v if $u \in N_D^+(v)$. When D is clear from the context, we drop it from the notation.

A *subgraph* of a (di)graph G is a (di)graph H with $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$, and with the property that the incidence function ψ_H associated with H is the restriction of ψ_G to $E(H)$. We say that H is *induced* if for every $u,v \in V(H)$, we have that $(u,v) \in E(H)$ whenever $(u,v) \in E(G)$. For a set $X \subseteq V(G)$, we write $D \setminus X$ for the digraph resulting from the deletion of X from D, and we denote by $G[X]$ the *subgraph of G induced by X*; that is, the induced subgraph of G with vertex set X. We write $H \subseteq G$ when H is a subgraph of G.

A *walk* in a (di)graph G is an alternating sequence W of vertices and edges that starts and ends with a vertex, and such that for every edge (u,v) in the walk, vertex u (resp. vertex v) is the element right before (resp. right after) edge (u,v) in W. If the first vertex in a walk is u and the last one is v, then we say this is a *walk from u to v*. A *path* is a (di)graph containing exactly a walk that contains all of its vertices and edges without repetition. If P is a path with $V(P) = \{v_1,\ldots,v_k\}$ and $E(P) = \{(v_i,v_{i+1}) \mid i \in [k-1]\}$, we say that v_1 is the *first* vertex of P, that v_k is the *last* vertex of P, and for $i \in [k-1]$ we say that v_{i+1} is the *sucessor in* P of v_i. If $P \subseteq G$, we say that P is a *path from* v_1 *to* v_k in G. A *cycle* is a (di)graph formed by a path containing at least two vertices together with an edge from its first to its last vertex. The *length* of a path or a cycle is the number of edges it contains. All paths and cyles mentioned henceforth, unless stated otherwise, are considered to be directed.

An *orientation* of an undirected graph G is a digraph D obtained from G by choosing an orientation for each edge $e \in E(G)$. The undirected graph G formed by ignoring the orientation of the edges of a digraph D is the *underlying graph* of D.

We say that an undirected graph G is *connected* if, for every pair of vertices $u, v \in V(G)$, there is a path from u to v in G. A component of G is maximal connected subgraph of G. A digraph D is *strongly connected* if, for every pair of vertices $u, v \in V(D)$, there is a walk from u to v and a walk from v to u in D. We say that D is *weakly connected* if the underlying graph of D is connected. A *separator* of D is a set $S \subsetneq V(D)$ such that $D \setminus S$ is not strongly connected. If $|V(D)| \geq k+1$ and k is the minimum size of a separator of D, we say that D is *k-strongly connected*. A *strong component* of D is a maximal induced subgraph of D that is strongly connected, and a *weak component* of D is a maximal induced subgraph of D that is weakly connected.

A *tree* is an undirected connected and acyclic graph. We also abbreviate acyclic digraphs with the acronym *DAG*. An *arborescence* R *with root* v is an orientation of a tree such that there is a path from v to every other vertex in $V(R) \setminus \{v\}$. If a vertex $u \in V(R)$ has out-degree zero, we say that u is a *leaf* of R.

An *independent set* in a (di)graph G is a set of pairwise non-adjacent vertices of G. A *clique* is a set of pairwise adjacent vertices of G. We say that a graph G is a *complete graph* if $V(G)$ is a clique and an *empty graph* if $V(G)$ is an independent set. A *tournament* is an orientation of a complete graph.

For $X, Y \subseteq V(D)$, an (X,Y)-*separator* is a set of vertices S such that there are no paths in $D \setminus S$ from any vertex in X to any vertex in Y. If $X = \{u\}$ we just write (u,Y)-separator instead of $(\{u\},Y)$-separator (omitting the braces), and do the same if $Y = \{v\}$. We make use Menger's Theorem [31] for digraphs.

Theorem 2.0.1 (Menger's Theorem [31])**.** *Let D be a digraph and $X, Y \subseteq V(D)$. Then the minimum size of an (X,Y)-separator in D equals the maximum number of pairwise internally vertex-disjoint paths from X to Y in D.*

For two positive integers a and b with $a \geq b$, the *Stirling number of the second kind* [32], denoted by $\mathsf{Stirling}(a,b)$, counts the number of ways to partition a set of a objects into b non-empty subsets, and is bounded from above by $\frac{1}{2}\binom{a}{b} \cdot b^{a-b}$.

2.1 Parameterized complexity

Are all NP-hard problem *equally* hard? Under the optics of approximation there are, for instance, hard problems for which we can compute solutions that are within a constant factor of an optimal solution, and those for which no algorithm with such a guarantee can exist unless $\mathsf{P} = \mathsf{NP}$. In some cases we can find a close-to-optimal solution to many instances of NP-hard problems by applying heuristic algorithms, albeit without a general guarantee of the approximation ratio.

Considerable effort is applied into optimizing exponential functions bounding the running time of algorithms solving NP-hard problems. As mentioned in Chapter 1, *parameterized complexity* offers another approach to deal with hard problems where the goal is to decide how tractable a given decision problem is with relation to a parameter carrying some information about instances of the problem. The first large study on parameterized complexity was done in the monograph by Downey and Fellows [33] and, for further references, we refer the reader to [6,34]. To further justify the results of this thesis, in this section we delve a littler deeper in the field of parameterized complexity, particularly on topics regarding tree-width and bidimensionality.

2.1.1 Formal definitions

A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$ where Σ is a fixed, finite alphabet. If (x,k) is an instance of a parameterized problem, k is called the *parameter*. For example, an instance of CLIQUE parameterized by the size of the solution is a pair (G,k) where G is the input graph and k is the parameter.

We say that L is *slice-wise polynomial* if there exists an algorithm $\mathcal{A}$ and two computable functions $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$ such that, given an instance $(x,k) \in \Sigma^* \times \mathbb{N}$, $\mathcal{A}$ correctly decides whether $(x,k) \in L$ in time bounded by $f(k) \cdot |(x,k)|^{g(k)}$. In this case, $\mathcal{A}$ is called an XP *algorithm* and we say that L is solvable in XP time. The complexity class containing all slice-wise polynomial problems is called XP. We also say that L is *fixed-parameter tractable* if there is an algorithm $\mathcal{A}$, a computable function $f : \mathbb{N} \to \mathbb{N}$, and a constant c such that $\mathcal{A}$

correctly decides whether an instance $(x,k) \in \Sigma^* \times \mathbb{N}$ is in L in time bounded by $f(k) \cdot |(x,k)|^c$. In this case, $\mathcal{A}$ is called an FPT algorithm and we say that L is solvable in FPT time. The complexity class containing all fixed-parameter tractable problems is called FPT. As examples, when parameterized by the size of the solution, CLIQUE is XP and VERTEX COVER is FPT.

There is significant difference between the running times of an XP an FPT algorithm for a given parameterized problem L, in favor of the latter, and thus the standard goal within the optics of parameterized complexity when dealing with such a problem L is to find an FPT algorithm for it or provide evidence that L is *not* FPT. Within parameterized problems, the W-hierarchy may be seen as the parameterized equivalent to the class NP of classical decision problems. A parameterized problem being $\mathsf{W}[1]$-*hard* can be seen as a strong evidence that this problem is *not* FPT. The canonical example of $\mathsf{W}[1]$-hard problem is CLIQUE parameterized by the size of the solution, and Downey and Fellows [8] showed that CLIQUE and INDEPENDENT SET are $\mathsf{W}[1]$-complete when parameterized by the size of the solution.

Parameterized reductions are used to transfer fixed-parameter tractability or hardness between parameterized problems. Namely, a parameterized reduction is an algorithm that, given an instance (x,k) of a parameterized problem L, runs in time $f(k) \cdot |x|^{\mathcal{O}(1)}$ and outputs an instance (x',k') of a parameterized problem L' such that $k' \leq g(k)$ for some computable function g and (x,k) is positive if and only if (x',k') is positive. For example, if L is $\mathsf{W}[1]$-hard and there is a parameterized reduction from L to L', then L' is also $\mathsf{W}[1]$-hard and thus unlikely to admit an FPT algorithm.

Most of the nautral NP-hard problems are equivalent to each other with respect to polynomial time reductions in the sense that are reductions from $\mathcal{P}$ to $\mathcal{Q}$ and from $\mathcal{P}$ to $\mathcal{Q}$, for every pair $\mathcal{P}, \mathcal{Q}$ of NP-complete problems. However, this does not seems to be the case within hard parameterized problems. For example, there is a parameterized reduction from CLIQUE to DOMINATING SET (both parameterized by the size of the solution), but the other direction seems to be unlikely. In fact, the latter is the canonical example of a $\mathsf{W}[2]$-hard problem, and it is $\mathsf{W}[2]$-complete. Thus there is a hierarchy within parameterized problems following

$$\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{W}[t] \subseteq \mathsf{XP}.$$

For an instance (x,k) of a parameterized decision problem L, a *kernelization algorithm* is an algorithm $\mathcal{A}$ that, in polynomial time, generates from (x,k) an equivalent instance (x',k') of L such that $|x|' + k' \leq f(k)$, for some computable function f. If $f(k)$ is bounded from above by a polynomial of the parameter, we say that L admits a *polynomial kernel*. We also say that L admits a kernel of *size* $f(k)$, and it is folklore that a parameterized problem admits a kernel if and only if it is FPT.

A *polynomial time and parameter reduction* (PPT) is similar to a parameterized reduction, but runs in polynomial time and also requires k' to be bounded from above by a polynomial on k. Polynomial time and parameter reductions can be used to show evidence that a parameterized problem is unlikely to admit a polynomial kernel: if L is unlikely to admit a polynomial kernel, then the existence of a PPT from L to L' is evidence that L' is also unlikely to admit a polynomial kernel.

See Figure 1 for an illustration of the hierarchy between parameterized problems.

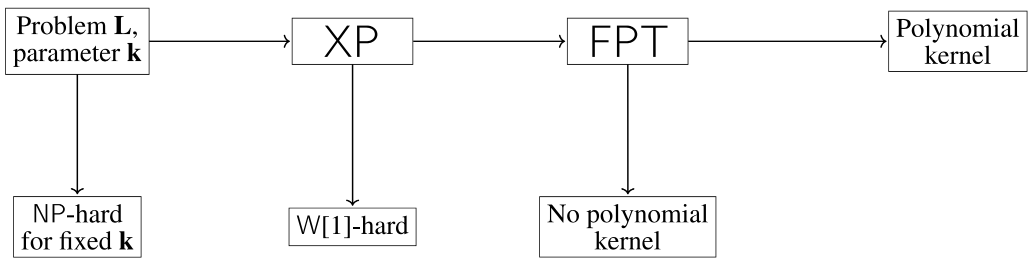


Figure 1 – Hierarchy of parameterized problems.

2.2 Tree-width

Most of the examples of parameterized problems we mentioned thus far are considering a parameter that is naturally related to some aspect of the solutions. CLIQUE parameterized by the size of the solution is W[1]-hard and thus there is little hope in finding an FPT algorithm for it under this parameterization. On the positive side, there are many hard problems that become tractable under parameterizations that, although not directly linked to solutions, measure some *structural* property of input instances.

Consider, for instance, the maximum degree $\Delta(G)$ of a graph G. If $\Delta(G) \leq k-2$ we can immediately say that the instance of CLIQUE is negative since every vertex in a clique of size k clearly has $k-1$ neighbors in the clique. If this is not the case notice that, when given a vertex v of G, we can check whether there is a clique of size k containing v by looking at each of the $2^{\Delta(G)}$ subsets of neighbors of v. This simple idea yields an $\mathcal{O}(2^{\Delta(G)} \cdot \Delta^2(G) \cdot n)$ algorithm to solve CLIQUE and thus this problem is FPT when parameterized by $\Delta(G)$ (and notice that this statement holds for *any* choice of k).

There is nothing special in this comparison between $\Delta(G)$ and k since, in general, not much can be said about instances of CLIQUE whose input graph has maximum degree at least $k-1$. The *tree-width* of a graph is more interesting than the maximum degree in this regard since not only many hard problems become tractable in graphs of bounded tree-width, but in many cases the existence of *blocking structures* certifying that the tree-width has to be large allows the development of FPT algorithms with subexponential dependence on this parameter, as well as other approaches to attack parameterized problems in graphs. Informally, the *tree-width* of a graph G measures how close G is to a tree.

Definition 2.2.1 (Tree decomposition). *A* tree decomposition $\mathcal{T}$ *of a graph G is a pair* $(T,\mathcal{X})$ *where T is a tree,* $\mathcal{X}$ *is a collection* $\{X_t \mid t \in V(T)\}$ *of* bags *of vertices of G, and*

(1) $X_1 \cup X_2 \cup \cdots \cup X_t = V(G)$*;*

(2) for every edge (u,v) *of G there is a bag* X_t *containing both u and v; and*

(3) for every $u \in V(G)$*, the set* $T_u = \{t \in V(T) \mid u \in X_t\}$ *induces a connected subgraph of T.*

The width of $\mathcal{T}$ *is the least integer k such that* $|X_t| \leq k+1$ *for every* $t \in V(T)$*. The* tree-width *of a graph G, denoted by* tw(G)*, is the least integer k such that G has a tree decomposition of width k.*

See Figure 2 for an example of a graph and a tree decomposition for it.

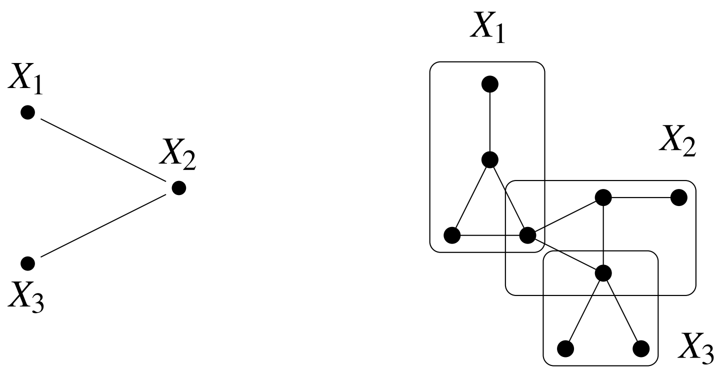


Figure 2 – An example of a graph (on the right) and a tree decomposition for it (on the left). Each set X_i is a bag of the decomposition.

It is easy to construct a tree decomposition $(T,\mathcal{X})$ of width one for a tree F. If $V(F)=1$, let $V(T)=\{t\}$, $\mathcal{X}=\{X_t\}$, and $X_t=V(F)$. Otherwise, start with $\mathcal{X}=\emptyset$ and add to it one bag X_e for each edge $e \in E(F)$ and one bag X_v for each $v \in V(F)$. Then, add to T the associated vertices t_e and t_v, respectively. Finally, add an edge between vertices t_e and t_v in T if and only if v is one of the endpoints of e. Since every graph containing at least one edge has tree-width at least one, we conclude that trees are exactly the connected graphs with tree-width at most one. With other simple observations, as seen in [6], one can show that cycles have tree-width 2 and that any graph containing a clique of size k has tree-width at least $k-1$.

Arnborg et al. [35] showed that in general it is NP-hard to compute the tree-width of a given graph G. On the positive side, Bodlaender showed that we can decide if $\mathsf{tw}(G) \leq k$, and construct a tree-decomposition for the input graph if this is the case, in FPT time [36].

Theorem 2.2.2 (Bodlaender [36])**.** *Let G be a graph and k be a positive integer. There is an algorithm running in time $\mathcal{O}(k^{\mathcal{O}(k^3)} \cdot n)$ that either produces a tree-decomposition of G of width at most k or correctly decides that $\mathsf{tw}(G) > k$.*

A classical result by Robertson and Seymour [37] shows that there is an approximation algorithm for tree-width with a better dependence on the parameter, but with a quadratic dependence on n.

Theorem 2.2.3 (Robertson and Seymour [37])**.** *Let G be a graph and k be a positive integer. There is an algorithm running in time $\mathcal{O}(8^k \cdot k^2 \cdot n^2)$ that either produces a tree-decomposition of G of width at most k or correctly decides that $\mathsf{tw}(G) > k$.*

As is the case with maximum degree, CLIQUE is FPT with relation to the tree-width of the input graph, as are the following problems: VERTEX COVER, DOMINATING SET, FEEDBACK VERTEX SET, CONNECTED DOMINATING SET, and many others. Moreover, Courcelle's Theorem [38] states that every problem that can be modeled with monadic second order logic admits an FPT algorithm when parameterized by the tree-width of the input graph.

We now briefly discuss the framework of *Bidimensionality* and, as an application, discuss how it can be used to provide subexponential FPT algorithms for parameterized problems in planar graphs. We introduce this powerful framework through simple examples connecting the main ideas used in it. We use the following two classical problems. A *vertex cover* of a graph G is set $X \subseteq V(G)$ such that there are no edges in $G \setminus X$.

LONGEST PATH

Input: A graph G and a positive integer k.

Output: A path of size at least k in G.

VERTEX COVER

Input: A graph G and a positive integer k.

Output: A vertex cover of G with size at most k.

We denote by $\ell\mathsf{p}(G)$ the size of a longest path in G and by $\mathsf{vc}(G)$ the minimum size of a vertex cover of G. It is not hard to see that a $(k \times k)$*-grid*, or a *grid of order k*, contains a path of size k^2 and a set of $\lfloor k^2/2 \rfloor$ independent edges. Such a set of edges in a graph is known as a *matching*. See Figure 3 for a drawing of a grid of order k, of a path, and of a matching in this grid.

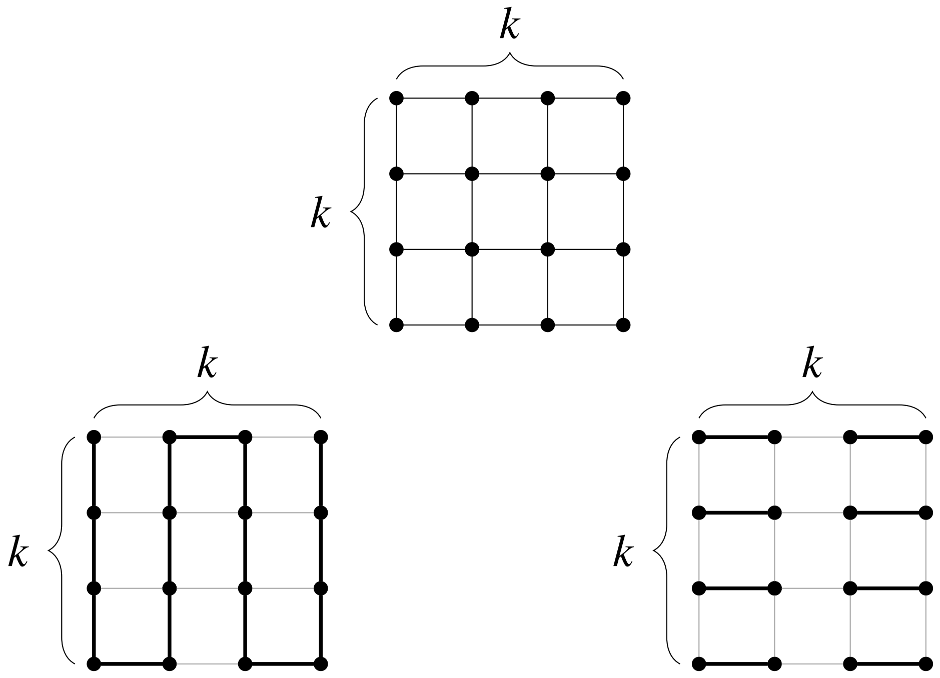


Figure 3 – A grid of order k (top), a path of size k^2 (bottom left) and a matching of size $\lfloor k^2/2 \rfloor$ (on the right) in a grid of order k.

Also notice that both LONGEST PATH and VERTEX COVER are closed with relation to minors: if G is a graph, H is a minor of G, and H contains a path of size k, for example, then clearly G also contains a path of size k since contractions can only decrease the size of a path. Thus the size of a longest path in G is at least the size of a longest path in any minor of G. Similarly, the size of a vertex cover of G has to be at least the size of a vertex cover of any minor of G. Adding those observations with the bounds on the size of longest paths and vertex covers of grids discussed in the previous paragraph, we obtain the following remark.

Remark 2.2.4. *Every graph G containing a grid of order k as a minor has $\ell\mathsf{p}(G) \geq k^2$ and $\mathsf{vc}(G) \geq \lfloor k^2/2 \rfloor$.*

The grid Theorem by Robertson and Seymour [5] states that there is a relationship between the tree-width of a graph G and the order of the largest grid minor of G. Robertson, Seymour, and Thomas [39] and Gu and Tamaki [40], this time with a improved bound, showed that this relationship is linear when G is planar. We state the version by Gu and Tamaki [40].

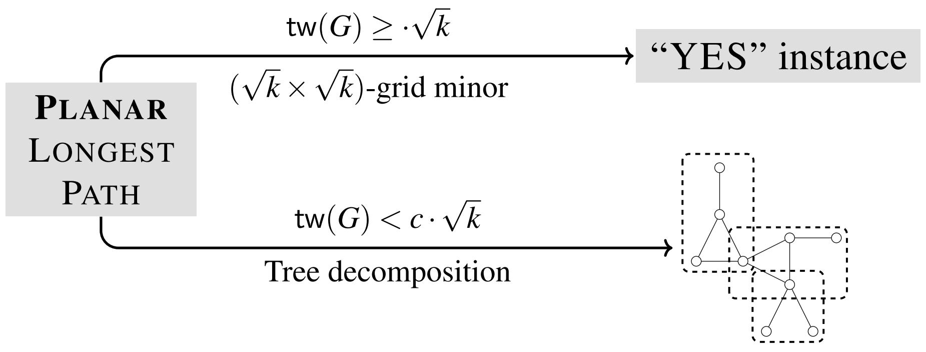
Theorem 2.2.5 (Planar excluded grid Theorem [40])**.** *Let G be a planar graph. If $\mathsf{tw}(G) \geq 9t/2$ then G contains a $(t \times t)$-grid minor. Furthermore, for every $\varepsilon > 0$ there exists an algorithm running in time $\mathcal{O}(n^2)$ that receives as input a planar graph G and a integer t and either outputs a tree decomposition of G with width at most $(9/2 + \varepsilon)t$ or constructs a minor model of a grid of order t in G.*

Finally, a classical results stated in [6, Chapters 7,11], for example, show that LONGEST PATH and VERTEX COVER are FPT when parameterized by the tree-width of the input graph.

Remark 2.2.6. LONGEST PATH *and* VERTEX COVER *are solvable in time $2^{\mathsf{tw}(G)} \cdot n^{\mathcal{O}(1)}$.*

Connecting the pieces. Let (G,k) be an instance of LONGEST PATH where G is planar. As done with CLIQUE and $\Delta(G)$ (the maximum degree of G) in the beginning of this section, we start by comparing the size k of solutions for LONGEST PATH with the tree-width $\mathrm{tw}(G)$ of the input graph. Applying Theorem 2.2.5 with inputs G and $\sqrt{k}$ we either obtain a $(\sqrt{k} \times \sqrt{k})$-grid minor H of G, or a tree decomposition of width at most $9\sqrt{k}/2$ of G. In the first case, by Remark 2.2.4 we conclude that $\ell\mathsf{p}(G) \geq k$ since $\ell\mathsf{p}(H) \geq k$ (see Figure 3 for an example) and thus the instance is positive and H is a certificate of that. In the second case, we apply Remark 2.2.6 to solve (G,k) in time $2^{\mathcal{O}(\sqrt{\mathsf{tw}(G)})} \cdot n^{\mathcal{O}(1)}$. Since we can decide which case occurs in polynomial time, we now have a "win/win" scenario where we either win because we find a large grid minor certifying that the instance is positive (or negative), or obtain a tree-decomposition and use it to solve the problem in subexponential time. See Figure 4 for an illustration of this scenario.

Thus connecting the pieces stated in this section (Remark 2.2.4, Theorem 2.2.5, and Remark 2.2.6) we conclude that there is an algorithm running in time $2^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ for the LONGEST PATH problem, and we can produce an algorithm with the same running time for the VERTEX COVER problem applying the same technique.



$\mathsf{tw}(G) \geq \cdot\sqrt{k}$
$(\sqrt{k} \times \sqrt{k})$-grid minor
"YES" instance
PLANAR LONGEST PATH
$\mathsf{tw}(G) < c \cdot \sqrt{k}$
Tree decomposition

Figure 4 – Visualizing LONGEST PATH and Bidimensionality. Here $c = 9/2$.

LONGEST PATH and VERTEX COVER are examples of *bidimensional* problems. Roughly speaking, a problem $\mathcal{Q}$ is *bidimensional* if it is minor-closed and if the size of a solution for $\mathcal{Q}$ in a grid grows quadratically in relation to the order of the grid. Thus any bidimensional problem that is solvable in time $2^{\mathcal{O}(\mathsf{tw}(G))} \cdot n^{\mathcal{O}(1)}$, for example, is solvable in time $2^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ in planar graphs.

2.3 Arboreal decompositions and directed tree-width

We now define guarded sets and arboreal decompositions of digraphs. From here on, we refer to oriented edges only, unless stated otherwise. D will always stand for a digraph, and G for an undirected graph. Unless stated otherwise, we define $n = |V(D)|$ and $m = |E(D)|$ when D is the input digraph of some algorithm.

Definition 2.3.1 (Z-guarded sets). *Let D be a digraph, $Z \subseteq V(D)$, and $S \subseteq V(D) \setminus Z$. We say that S is* Z-guarded *if there is no directed walk in $D \setminus Z$ with first and last vertices in S that uses a vertex of $D \setminus (Z \cup S)$. For an integer $w \geq 0$, we say that S is* w-guarded *if S is Z-guarded for some Z with $|Z| \leq w$.*

That is, informally speaking, a set S is Z-guarded if whenever a walk starting in S leaves S, it is impossible to come back to S without visiting a vertex in Z. See Figure 5 for an illustration of a Z-guarded set. If a set S is Z-guarded, we may also say that Z is a *guard* for S. We remark that in [24], the authors use the terminology of *Z-normal* sets instead of Z-guarded sets.

Given the enormous success achieved by applications based on width parameters in undirected graphs, it is no surprise that there is interest in finding similar definitions for digraphs. Johnson et al. [24] proposed an analogous measure for tree-width in the directed case. The *directed tree-width* of a digraph measures its distance to being a DAG, and an *arboreal*

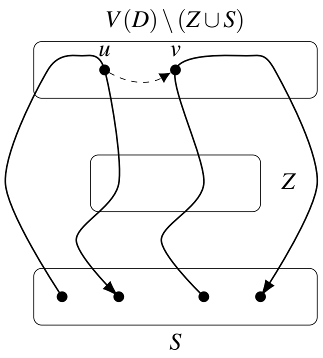


Figure 5 – A Z-guarded set S. The dashed line indicates that there is no path from u to v in $V(D)\setminus(Z\cup S)$.

decomposition exposes a (strong) connectivity measure of a digraph. Reed [41] gave an intuition for the similarities between the undirected and directed cases.

Let R be an arborescence, $r \in V(R)$, $e \in E(R)$, and r' be the head of e. We say that $r > e$ if there is a path from r' to r in R (notice that this definition implies that $r' > e$). We also say that $e \sim r$ if r is the head or the tail of e. To define the directed tree-width of directed graphs, we first need to introduce arboreal decompositions.

Definition 2.3.2 (Arboreal decomposition)**.** *An* arboreal decomposition β *of a digraph* D *is a triple* $(R,\mathcal{X},\mathcal{W})$ *where* R *is an arborescence,* $\mathcal{X} = \{X_e : e \in E(R)\}$, $\mathcal{W} = \{W_r : r \in V(R)\}$, *and* $\mathcal{X},\mathcal{W}$ *are collections of sets of vertices of* D *(called* bags*) such that*

(i) $\mathcal{W}$ *is a partition of* $V(D)$ *into non-empty sets, and*

(ii) *if* $e \in E(R)$*, then* $\bigcup\{W_r : r \in V(R) \text{ and } r > e\}$ *is* X_e*-guarded.*

We also say that r *is a* leaf *of* $(R,\mathcal{X},\mathcal{W})$ *if* r *has out-degree zero in* R*.*

The left hand side of Figure 6 contains an example of a digraph D, while the right hand side shows an arboreal decomposition for it. In the illustration of the arboreal decomposition, squares are guards X_e and circles are bags of vertices W_r. For example, consider the edge $e \in E(R)$ with $X_e = \{b,c\}$ from the bag W_1 to the bag W_2. Then $\bigcup\{W_r : r \in V(R) \text{ and } r > e\} = V(D)\setminus\{a\}$ and, by item **(ii)** described above, this set must be $\{a\}$-guarded since $X_e = \{a\}$. In other words, there cannot be a walk in $D\setminus\{b,c\}$ starting and ending in $\{a\}$ using a vertex of $V(D)\setminus\{a\}$. This is true in D since every path reaching $\{a\}$ from the remaining of the graph must do so through vertices b or c. The reader is encouraged to verify the same properties for the other guards in the decomposition.

Definition 2.3.3 (Nice arboreal decompositions)**.** *We say that an arboreal decomposition* $(R,\mathcal{X},\mathcal{W})$ *of a digraph* D *is* nice *if*

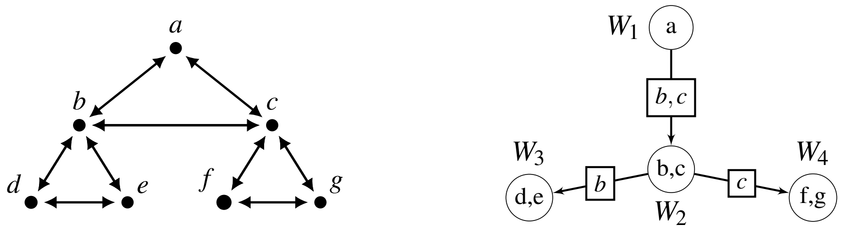


Figure 6 – A digraph D and an arboreal decomposition of D of width two. A bidirectional edge is used to represent a pair of edges in both directions.

(iii) *for every $e \in E(R)$, $\bigcup\{W_r : r \in V(R), r > e\}$ induces a strong component of $D \setminus X_e$, and*

(iv) *if $r \in V(R)$ and $r_1, \dots, r_\ell$ are the out-neighbors of r in R, then*

$$\left(\bigcup_{i\in[\ell]} W_{r_i}\right) \cap \left(\bigcup_{e\sim r} X_e\right) = \emptyset.$$

Definition 2.3.4 (Directed tree-width)**.** *Let $(R, \mathcal{X}, \mathcal{W})$ be an arboreal decomposition of a digraph D. For a vertex $r \in V(R)$, we denote by $\mathsf{width}(r)$ the size of the set $W_r \cup (\bigcup_{e\sim r} X_e)$. The* width *of $(R, \mathcal{X}, \mathcal{W})$ is the least integer k such that, for all $r \in V(R)$, $\mathsf{width}(r) \leq k+1$. The* directed tree-width *of D, denoted by $\mathsf{dtw}(D)$, is the least integer k such that D has an arboreal decomposition of width k.*

We remark that DAGs have directed tree-width zero.

If G is an undirected graph and D the digraph obtained from G by replacing every edge of G with two directed edges in opposite directions then, as shown by Johnson et al. [24], the tree-width of G is equal to the directed tree-width of D. Thus, deciding if a digraph D has directed tree-width at most k, for a given integer k, is NP-complete since deciding if the tree-width of an undirected graph is at most k is an NP-complete problem [35].

Similarly to the undirected case, some hard problems become tractable when restricted to digraphs of bounded directed tree-width. For example, Johnson et al. [24] showed that the Directed Disjoint Paths (DDP) problem, which Fortune et al. [2] showed to be NP-hard even when the goal is to connect $k = 2$ pairs of vertices only, is XP with parameters k and $\mathsf{dtw}(D)$. A similar approach given in [24] can be applied to the Hamilton Path and Hamilton Cycle problems, Hamilton Path with Prescribed Ends, and others. However, Slivkins [3] proved that DDP with parameter k is W[1]-hard even when restricted to DAG. As DAGs have directed tree-width zero, there is little hope for the existence of an FPT algorithm for this parameterization of DDP in digraphs of bounded directed tree-width. As another example of application, a Courcelle-like theorem for directed tree-width is shown in [42], but running in XP time.

We now formally define cylindrical grids, butterfly contractions, butterfly minors, and some blocking structures for large directed tree-width.

Definition 2.3.5 (Cylindrical grid). *A* cylindrical grid of order *k is a digraph formed by the union of k disjoint cycles $C_1,\dots,C_k$ and $2k$ disjoint paths P_1, P_2, ..., P_{2k} where*

1. *for $i \in [k], V(C_i) = \{v_{i,1}, v_{i,2}, \dots, v_{i,2k}\}$ and $E(C_i) = \{(v_{i,j}, v_{i,j+1} \mid j \in [2k-1])\} \cup \{(v_{i,2k}, v_{i,1})\}$,*
2. *for $i \in \{1,3,\dots,2k-1\}$, $E(P_i) = \{(v_{1,i}, v_{2,i}), (v_{2,i}, v_{3,i}), \dots, (v_{k-1,i}, v_{k,i})\}$, and*
3. *for $i \in \{2,4,\dots,2k\}$, $E(P_i) = \{(v_{k,i}, v_{k-1,i}), (v_{k-1,i}, v_{k-2,i}), \dots, (v_{2,i}, v_{1,i})\}$.*

In other words, path P_i is oriented from the first circle to the last one if i is odd, and the other way around if i is even. Furthermore, every vertex of a cylindrical grid occurs in the intersection of a path and a cycle. See Figure 7 for an example of a cylindrical grid of order $k = 4$.

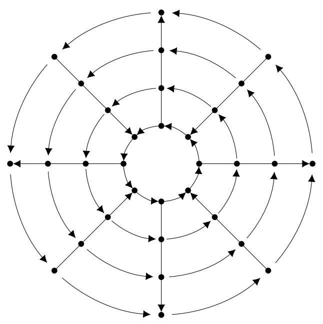


Figure 7 – A cylindrical grid of order $k = 4$.

Definition 2.3.6 (Butterfly contraction and butterfly minors). *Let D be a digraph. An edge e from u to v of D is* butterfly contractible *if e is the only outgoing edge of u or the only incoming edge of v. By* butterfly contracting *e in D, we obtain a digraph D' with vertex set $V(D') = V(D) \setminus \{u,v\} \cup x_{u,v}$, where $x_{u,v}$ is a new vertex, and $E(D') = E(D) \setminus \{e\}$. Every incidence of an edge $f \in E(D')$ to u or v in D becomes an incidence to $x_{u,v}$ in D'. If D' is generated from a subgraph of D by a series a butterfly contractions, we say that D' is a* butterfly minor *of D.*

Notice that, in the above definition, the newly introduced vertex $x_{u,v}$ has in D' the same neighbors of u and v in D (minus u and v). It is not hard to see that butterfly contractions cannot generate any new paths and that there is no such guarantee if no restrictions are imposed on which edges of a digraph can be contracted. See Figure 8 for an example of this.

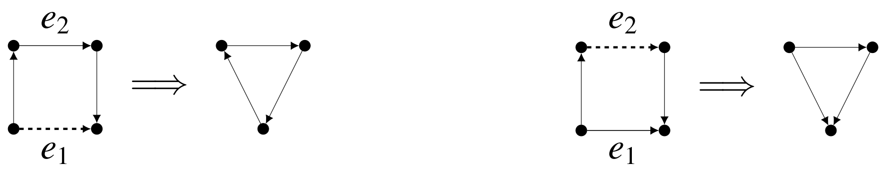


Figure 8 – Butterfly contractions preserves separations. In each example the dashed edge is contracted to generate the digraph on the right. The edge e_1 is *not* butterfly contractible.

2.4 List of problems

We provide here the definition of the problems that are mentioned in this thesis. For the sake of completeness, we also include the following definitions, which we state again in sections 3.1 and 4.1 when they are needed.

Definition 2.4.1 ((T,r)-balanced separators and (k,r)-linked sets)**.** *Let D be a digraph, $T \subseteq V(D)$, and r be a positive integer. A (T,r)-*balanced separator *is a set of vertices Z such that every strong component of $D \setminus Z$ contains at most r vertices of T. If the minimum size of a (T,r)-balanced separator is at least $k+1$, we say that T is (k,r)*-linked.

Definition 2.4.2 (Requests and satisfying collections)**.** *Let D be a digraph and $\mathcal{P}$ be a collection of paths of D. A* request *in D is an ordered pair of vertices of D. For a request set $I = \{(s_1,t_1),(s_2,t_2),\dots,(s_k,t_k)\}$, we say that the vertices $\{s_1,s_2,\dots,s_k\}$ are* source *vertices and that $\{t_1,t_2,\dots,t_k\}$ are* target *vertices, and we refer to them as $S(I)$ and $T(I)$, respectively. We say that $\mathcal{P}$* satisfies *I if $\mathcal{P} = \{P_1,\dots,P_k\}$ and P_i is a path from s_i to t_i, for $i \in [k]$.*

We remark that a requests set may contain many copies of the same pair.

CLIQUE

Input: A digraph D and a non-negative integer k.

Output: A clique of size at least k in D.

LONGEST PATH

Input: A graph G and a positive integer k.

Output: A path of size at least k in G.

VERTEX COVER

Input: A graph G and a positive integer k.

Output: A vertex cover of G with size at most k.

DIRECTED DISJOINT PATHS

Input: A digraph D, a request set I of size k.

Output: A collection of pairwise vertex-disjoint paths $\mathcal{P}$ satisfying I.

DIRECTED DISJOINT PATHS WITH CONGESTION (DDPC)

Input: A digraph D, a request set I of size k, and a non-negative integer c.

Output: A collection of paths $\mathcal{P}$ satisfying I such that each vertex of D occurs in at most c paths of the collection.

DISJOINT ENOUGH DIRECTED PATHS (DEDP) (Page 67)

Input: A digraph D, a request set I of size k, and two non-negative integers c and s.

Output: A collection of paths $\mathcal{P}$ satisfying I such that at most c vertices of D occur in at least $s+1$ paths of $\mathcal{P}$ and all other vertices of D occur in at most s paths of $\mathcal{P}$.

INDEPENDENT SET

Input: A digraph D and a non-negative integer k.

Output: An independent set of size at least k in D.

LINEAR VERTEX CUT (Page 47)

Input: A digraph D, a collection of terminal sets $\mathcal{T}$, with $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$, where $T_i \subseteq V(D)$ for $i \in [k]$, and an integer $s \geq 0$.

Output: A set of vertices $Z \subseteq V(D)$ with $|Z| \leq s$ such that there are no paths in $D \setminus Z$ from T_i to T_j, for $1 \leq i < j \leq k$.

LINEAR EDGE CUT

Input: A digraph D, a collection of terminal sets $\mathcal{T}$, with $\mathcal{T} = \{T_1, T_2, \dots, T_k\}$, where $T_i \subseteq V(D)$ for $i \in [k]$, and an integer $s \geq 0$.

Output: A set of edges $Z \subseteq V(D)$ with $|Z| \leq s$ such that there are no paths in $D \setminus Z$ from T_i to T_j, for $1 \leq i < j \leq k$.

BALANCED SEPARATOR (Page 46)

Input: A digraph D, a set $T \subseteq V(D)$ of size k, and two non-negative integers r and s.

Output: A (T,r)-balanced separator Z with $|Z| \leq s$, if it exists.

STEINER NETWORK

Input: A digraph D, a request set of size k, and a non-negative integer c.

Output: A collection of paths $\mathcal{P}$ satisfying I such that $|\bigcup_{P \in \mathcal{P}} V(P)| \leq c$.

3 ADAPTING THE DIRECTED GRID THEOREM INTO AN FPT ALGORITHM

Width parameters can be seen as an estimation of how close a given graph is to a typical structure. For example, the *tree-width* of a graph, a parameter of particular interest in the literature, measures how tightly a graph can be approximated by a tree. A *tree decomposition* of a graph G with bounded tree-width shows how one can place the vertices of the original graph into "bags" of bounded size which, in turn, can be arranged as the vertices of a tree T such that the intersection between adjacent bags in T are separators in G. Thus, a tree decomposition exposes a form of global connectivity measure for graphs: as only a bounded number of vertices can be placed in each bag, many small separators can be identified through the decomposition. The tree-width of graphs was first introduced by Bertele and Brioschi [15], then again by Halin [16], and finally reintroduced by Robertson and Seymour [5]. For a survey on the subject, we refer the reader to [43].

A number of hard problems can be efficiently solved in graphs of bounded tree-width, either by making use of classical algorithmic techniques like dynamic programming, or by making use of Courcelle's Theorem [38]. Applications of algorithms based on tree decompositions range from frequency allocation problems to the TRAVELING SALESMAN problem [44, 45].

It is natural to ask what can be said of a graph with large tree-width. One of the most relevant results in structural graph theory states that undirected graphs with large tree-width contain large grid minors. More precisely, the Grid Theorem by Robertson and Seymour [5] states that there is a function $f : \mathbb{N} \to \mathbb{N}$ such that every graph of tree-width at least $f(k)$ contains a $(k \times k)$-grid as a minor. Recently, Chekuri and Chuzhoy [18] gave a polynomial function $f(k)$ for this result, which was further improved by Chuzhoy and Tan [19].

Sometimes, large tree-width (and therefore, the existence of a large grid minor) implies that we are actually working with a positive instance of a particular problem. Demaine et al. [21] gave a framework that generates FPT algorithms for many such problems, known as *bidimensional* problems. This list includes VERTEX COVER, FEEDBACK VERTEX SET, LONGEST PATH, MINIMUM MAXIMAL MATCHING, DOMINATING SET, EDGE DOMINATING SET, and many others. This seminal work is currently known as *Bidimensionality* [20].

Another application of the Grid Theorem is in the *irrelevant vertex* technique, introduced by Robertson and Seymour [1, 22, 23] to solve the DISJOINT PATHS problem. The goal is to show that every instance whose input graph violates a set of conditions contains a vertex that is "irrelevant", that is, a vertex whose removal generates an equivalent instance of the problem. This leads to an iterative algorithm, reducing the problem to a smaller instance, until it satisfies sufficient conditions for its tractability. This technique was used to provide an FPT algorithm for DISJOINT PATHS parameterized by the number k of requests, and a many other problems (cf. for instance [46,47]). For the directed case, Cygan et al. [29] used a similar technique to provide an FPT algorithm with the same parameter for the DIRECTED DISJOINT PATHS (DDP) problem in planar digraphs.

A result analogous to the Grid Theorem for digraphs was conjectured by Johnson et al. [24] and Reed [41], and recently proved by Kawarabayashi and Kreutzer [4][1], after having proved it for digraphs with forbidden minors [48][2]. Namely, it is shown in [4] that there is a function $f : \mathbb{N} \to \mathbb{N}$ such that every digraph of directed tree-width (see Section 2.3) at least $f(k)$ contains a cylindrical grid (see Figure 7) of order k as a butterfly minor. Recently, Hatzel et al. [50] proved that the function f can be made polynomial in *planar* digraphs.

The Directed Grid Theorem has found many applications. For instance, Amiri et al. [51] proved that a strongly connected digraph H has the Erdős-Pósa property if and only if H is a butterfly minor of some cylindrical grid of sufficiently large order. Additionally, the authors showed that for every fixed strongly connected digraph H satisfying those conditions and every fixed integer k, there is a polynomial-time algorithm that either finds k disjoint (butterfly or topological) models of H in a digraph D or a set $X \subseteq V(D)$ of size bounded by a function of k such that $D \setminus S$ does not contain a model of H.

Edwards et al. [52] applied some results used in the proof of the Directed Grid Theorem [4] to provide an algorithmic result for the DIRECTED DISJOINT PATHS WITH CONGESTION problem, or DDPC-c if we want to specify the congestion. Namely, they showed that DDPC-2 is XP with parameter k when restricted to $(36k^3 + 2k)$-strongly connected digraphs. Kawarabayashi and Kreutzer [4] mention that the Directed Grid Theorem can be used to provide, for fixed k, an algorithm running in polynomial time that either finds a solution for DDPC-4 or concludes that no set of pairwise vertex-disjoint paths satisfying the requests exists. Although Chekuri et al. [53] could not use the Directed Grid Theorem since the bound on $f(k)$ (mentioned above) is larger than required, they build on the ideas used in [49] to produce their own version of the Directed Grid Theorem for planar digraphs.

[1] The full version of [4] is available at `https://arxiv.org/abs/1411.5681`.

[2] In an unpublished manuscript from 2001 [49], Johnson, Robertson, Seymour and Thomas gave a proof of this result for planar digraphs.

A *bramble* $\mathcal{B}$ in a digraph D is a collection of strongly connected induced subgraphs of D such that every two elements of $\mathcal{B}$ either intersect or there is an edge from the first to the second and vice-versa. A *haven* of order k is a function β assigning to every set $Z \subseteq V(D)$ with $|Z| \leq k-1$ the non-empty vertex set of a strong component of $D \setminus Z$ in such way that $\beta(Z) \subseteq \beta(Z')$ whenver $Z' \subseteq Z$ (those definitions are also included in Section 3.1).

The proof of the Directed Grid Theorem by Kawarabayashi and Kreutzer [4] is constructive. Namely, the authors start from a result by Johnson et al. [24] stating that, given a digraph D and an integer parameter k, outputs, in XP time, either an arboreal decomposition of D of width at most $3k-2$ or a haven of order k. Thus, if D has directed tree-width at least $3k-1$, they obtain a haven of order k. From this haven, they obtain a bramble $\mathcal{B}$ of order k and size $|V(D)|^{\mathcal{O}(k)}$. Finally, from $\mathcal{B}$ they find a path P containing a well-linked set A of size roughly $\sqrt{k}$ in XP time with parameter k. We remark that the bound on the running time of those algorithms depends on the size of $\mathcal{B}$ since, in general, one must test whether $X \cap V(B) \neq \emptyset$ for each $B \in \mathcal{B}$ to check if a given set $X \subseteq V(D)$ is a hitting set of $\mathcal{B}$. The remainder of the proof of the Directed Grid Theorem [4] runs in FPT time, with parameter k.

Our approach, results, and techniques. By making local changes to the proofs by Johnson et al. [24] and Kawarabayashi and Kreutzer [4], we show that there is an FPT algorithm that, given a digraph D and an integer k, either constructs an arboreal decomposition of D of width at most $3k-2$, or finds a path P in D containing a well-linked set A of size roughly $\sqrt{2k}$. Together with the remainder of the proof of the Directed Grid Theorem given in [4], our results yield an FPT algorithm that either constructs an arboreal decomposition of width at most $f(k)$ or a cylindrical grid of order k as a butterfly minor of D.

In Section 3.1 we give the necessary definitions to formally state the main contributions of this chapter. In Section 3.2, we give an FPT algorithm that, given a digraph D and parameter k, outputs either an arboreal decomposition of D of width at most $3k-2$ or a haven $\mathcal{H}$ of order k, improving the result by Johnson et al. [24]. This result also shows that the size of a special kind of vertex separator, known as *balanced separator* [54], of some set $T \subseteq V(D)$ is intrinsically connected to the directed tree-width of D, similarly to the undirected case (see, for example, [55, Chapter 11]).

We acknowledge that a sketch of a proof of a similar result, with approximation factor of $5k+10$, is given in [54, Theorem 9.4.4]. In their proof, the authors mention how to compute a weaker version of balanced separators for a given set $T \subseteq V(D)$ in FPT time with parameter $|T|$, and the increase on the approximation factor they guarantee is a consequence of this relaxation. As a tool towards our FPT approximation algorithm for directed tree-width, we show in Section 3.2 how to compute a generalized version of balanced separators in FPT time with the same parameter. We make use of an algorithm by Erbacher et al. [56] for a variation of the MULTICUT problem for digraphs, named as MULTICUT WITH LINEARLY ORDERED TERMINALS by the authors.

In Section 3.3, we show how to use our algorithm for the generalized version of balanced separators for finding hitting sets for a specific bramble $\mathcal{B}$ of order k that naturally occurs in digraphs of directed tree-width at least $3k-1$ in FPT time with parameter k. The running time of our algorithm to find hitting sets of $\mathcal{B}$ does not depend on the *size* of $\mathcal{B}$, but only on its *order*. We remark that, in this particular case, we can decide if a given set $X \subseteq V(D)$ is a hitting set of $\mathcal{B}$ in polynomial time. This is an easy observation that also holds for the bramble used in the proof of the Directed Grid Theorem [4]. Finally, in Section 3.4, we use $\mathcal{B}$ and our algorithm to find hitting sets of $\mathcal{B}$ to find a path P containing a well-linked set A of order roughly $\sqrt{2k}$ in FPT time with parameter k.

A roadmap of the aforementioned algorithm is given in Figure 9. We mark by a dashed arc the steps of [4] which are already FPT and do not need to be adapted. All others edges represent steps that we adapt in this thesis.

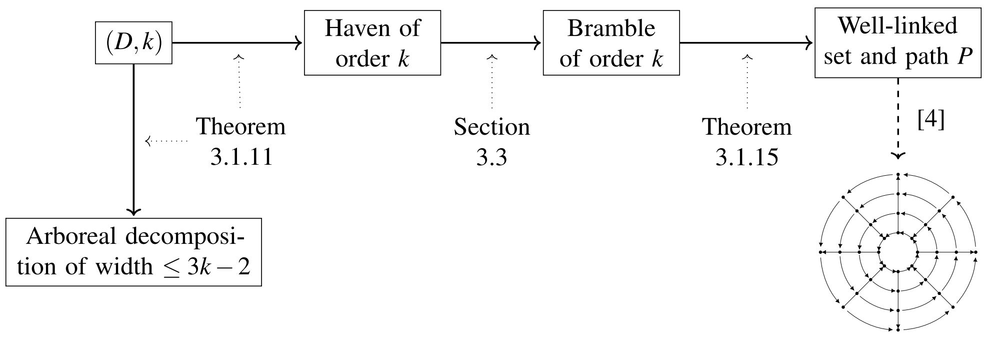


Figure 9 – Sketch of the algorithm used in the proof of the Directed Grid Theorem [4].

We conclude this chapter in Section 3.5 with some remarks and potential algorithmic applications of our results.

3.1 Preliminaries

In this section we give the definitions relevant to this chapter, mention some known results, and present a more detailed discussion of our main contributions.

Definition 3.1.1 (Well-linked sets)**.** *Let D be a digraph and $A \subseteq V(D)$. We say that A is* well-linked *in D if, for all disjoint $X, Y \subseteq A$ with $|X| = |Y|$, there are $|X|$ vertex-disjoint paths from X to Y in D. The* order *of a well-linked set A is $|A|$. We denote by $\mathsf{wlink}(D)$ the size of a largest well-linked set in D.*

Definition 3.1.2 (Havens in digraphs)**.** *Let D be a digraph. A* haven of order *k in D is a function β assigning to every set $Z \subseteq V(D)$, with $|Z| \leq k-1$, the non-empty vertex set of a strong component of $D \setminus Z$ in such way that if $Z' \subseteq Z \subseteq V(D)$ then $\beta(Z) \subseteq \beta(Z')$. The* haven number *of a digraph D, denoted by $\mathsf{hn}(D)$, is the maximum k such that D admits a haven of order k.*

A k-strongly connected digraph, for example, admits a haven of order k: it suffices to choose $\beta(Z) = V(D) \setminus Z$ for any $Z \subseteq V(D)$ with $|Z| \leq k-1$. See Figure 10 for an example of a 3-strongly connected digraph and an illustrates the defining property of havens.

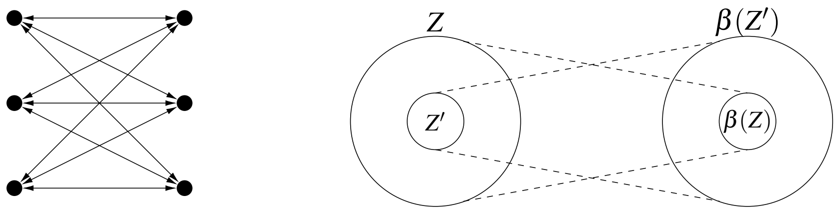
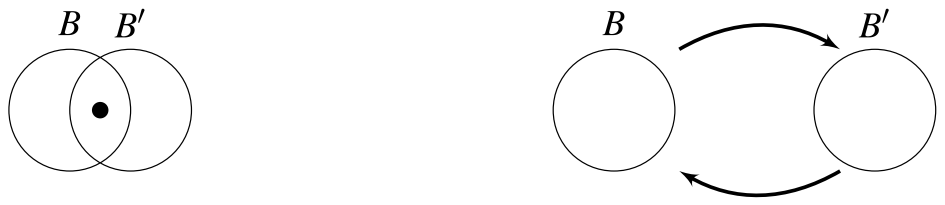


Figure 10 – Example of a 3-strongly connected digraph, and an illustration of the haven property. On the left, a bidirectional edge is used to represent a pair of edges in both directions.

Definition 3.1.3 (Brambles in digraphs)**.** *A* bramble *$\mathcal{B} = \{B_1, \ldots, B_\ell\}$ in a digraph D is a family of strongly connected subgraphs of D such that if $\{B, B'\} \subseteq \mathcal{B}$ then $V(B) \cap V(B') \neq \emptyset$ or there are edges in D from $V(B)$ to $V(B')$ and from $V(B')$ to $V(B)$. A* hitting set *of a bramble $\mathcal{B}$ is a set $C \subseteq V(D)$ such that $C \cap V(B) \neq \emptyset$ for all $B \in \mathcal{B}$. The* order *of a bramble $\mathcal{B}$, denoted by $\mathsf{ord}(\mathcal{B})$, is the minimum size of a hitting set of $\mathcal{B}$. The* bramble number *of a digraph D, denoted by $\mathsf{bn}(D)$, is the the maximum k such that D admits a bramble of order k.*

There is a direct relation between the haven number and the tree-width of undirected graphs. A haven in an undirected graph is defined similarly: the function β retains all its



(a) $V(B) \cap V(B') \neq \emptyset$. (b) There is an edge from B to B' and vice-versa.

Figure 11 – Illustration of the definining properties of brambles in digraphs. If B and B' are elements of a bramble, either (a) or (b) occurs.

properties, but mapping sets of at most $k-1$ vertices to components of the graph resulting from the deletion of those vertices.

Proposition 3.1.4 (Seymour and Thomas [57]). *Let G be an undirected graph and $k \geq 1$ be an integer. Then G has a haven of order k if and only if its tree-width is at least $k-1$.*

For digraphs, only one implication of the previous result is known to be true.

Proposition 3.1.5 (Johnson et al. [24]). *Let D be a digraph and k be a non-negative integer. If D has a haven of order k, then $\mathsf{dtw}(D) \geq k-1$.*

For the reverse direction of Proposition 3.1.5, only an approximate version is known.

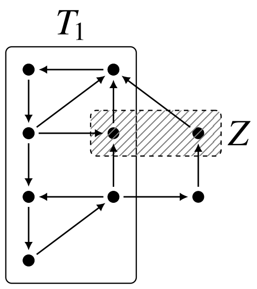
Proposition 3.1.6 (Johnson et al. [24]). *Let D be a digraph and k be a positive integer. If $\mathsf{dtw}(D) \geq 3k-1$ then D admits a haven of order k.*

Finally, the following two lemmas show that brambles of large order and large well-linked sets are obstructions to small directed tree-width. The proof of the first lemma can be done by converting brambles into havens and back. For the second lemma, it is sufficient to show that any minimum hitting set of a bramble of order k is well-linked and to extract a bramble of order k from a well-linked set of order $4k+1$. The proofs are simple and can be found, for example, in [58, Chapter 6].

Lemma 3.1.7. *Let D be a digraph. Then $\mathsf{bn}(D) \leq \mathsf{hn}(D) \leq 2\mathsf{bn}(D)$.*

Lemma 3.1.8. *Let D be a digraph. Then $\mathsf{bn}(D) \leq \mathsf{wlink}(D) \leq 4\mathsf{bn}(D)$.*

The proof of Proposition 3.1.6 given in [24] yields an XP algorithm that correctly states that D has a haven of order k or produces an arboreal decomposition of D of width at most $3k-2$. Furthermore, although not explicitly mentioned in the paper, this algorithm actually produces a nice (as in Definition 2.3.3) arboreal decomposition for D, and can be used



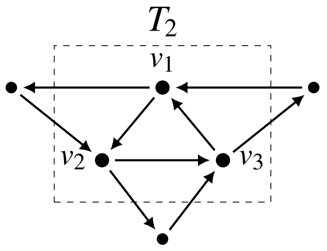


Figure 12 – Examples of balanced separators. On the left, Z is a $(T_1, 3)$-balanced separator, and T_1 is $(3, 3)$-linked. On the right, each square vertex v_i with $i \in [3]$ constitutes a $(T_2, 1)$-balanced separator.

as a procedure that, given a digraph D' such that $\mathsf{dtw}(D') \leq k - 2$, generates a nice arboreal decomposition for D' of width at most $3k - 2$. At each iteration, the algorithm tests whether the strong components intersecting a given set $T \subseteq V(D)$ with $|T| \leq 2k - 1$ can be separated into parts containing at most a small portion of T. Namely, the algorithm tests whether there is a set $Z \subseteq V(D)$ with $|Z| \leq k - 1$ such that every strong component of $D \setminus Z$ contains at most $k - 1$ vertices of $T \setminus Z$. Such a set Z is known as a *balanced separator* for T as defined, for instance, in [54, Chapter 9]. In this paper we consider a generalization of such sets where we can choose how many vertices of T each strong component of $D \setminus Z$ can have. We remind the reader of the following definition.

Definition 2.4.1 ((T, r)-balanced separators and (k, r)-linked sets)**.** *Let D be a digraph, $T \subseteq V(D)$, and r be a positive integer. A (T, r)-*balanced separator *is a set of vertices Z such that every strong component of $D \setminus Z$ contains at most r vertices of T. If the minimum size of a (T, r)-balanced separator is at least $k + 1$, we say that T is (k, r)-*linked.

If $r = \lfloor |T|/2 \rfloor$, (T, r)-balanced separators are exactly T-balanced separators in the classical sense as defined, for instance, in [54, Chapter 9]. If D admits a (T, r)-balanced separator Z, we know that we can split $T \setminus Z$ into small strongly connected parts which are guarded by Z. See Fig. 12 for an illustration of a (T, r)-balanced separator Z. A DAG, for instance, admits a $(T, 1)$-balanced separator (the empty set) for any $T \subseteq V(D)$ since every strong component of a DAG is formed by a single vertex.

Deciding whether a digraph D admits a $(T, \lfloor |T|/2 \rfloor)$-balanced separator is a key ingredient for the algorithm given by Johnson et al. [24]. Moreover, the cost of this procedure has the largest impact on the running time of their algorithm: it is the only step which is (originally) done in XP time, while the remaining parts of the algorithm can be done in polynomial time. In Section 3.2.2, we use of a variation of the MULTICUT problem introduced in [56] to show

how to find (T,r)-balanced separators in FPT time with parameter $|T|$, if any exists with size bounded from above by an integer s with $s \leq |T|-1$. In our first main contribution, we use this result to improve on the algorithm for arboreal decompositions given in [24]. Namely, we prove the following.

Theorem 3.1.9. *Let D be a digraph and k be a non-negative integer. There is an algorithm running in time $2^{\mathcal{O}(k\log k)} \cdot n^{\mathcal{O}(1)}$ that either produces a nice arboreal decomposition of D of width at most $3k-2$ or outputs a $(k-1,k-1)$-linked set T with $T = 2k-1$.*

It is also not hard to see how to use (k,r)-linked sets to construct havens. The following lemma is a generalization of a result shown as part of the proof of [24, 3.3].

Lemma 3.1.10. *Let D be a graph, $T \subseteq V(D)$ with $|T| = s$, and $r \geq \lfloor s/2 \rfloor$. If T is (k,r)-linked then D admits a haven of order $k+1$.*

Proof. By hypothesis, it holds that, for every set $Z \subseteq V(D)$ with $|Z| \leq k$, there is a strong component C of $D \setminus Z$ such that $|V(C) \cap T| \geq r+1$. Let $\beta(Z) = V(C)$. We claim that β is a haven of order $k+1$ in D. It suffices to show that if $Z' \subseteq Z$, then $\beta(Z) \subseteq \beta(Z')$. Notice that $\beta(Z)$ induces a strongly connected subgraph of D and is disjoint from Z', since it is disjoint from Z, and thus all paths in the graph induced by $\beta(Z)$ are in $D \setminus Z'$. Furthermore, since $|T| = s$ and $r \geq \lfloor s/2 \rfloor$, we have $\beta(Z) \cap \beta(Z') \neq \emptyset$ and the result follows as $\beta(Z')$ is a strong component of $D \setminus Z'$, which is a supergraph of $D \setminus Z$, and thus it must contain completely the strongly connected subgraph induced by $\beta(Z)$. $\square$

Applying this lemma on a $(k-1,k-1)$-linked set T with $|T| = 2k-1$ we obtain a haven of order k and therefore we can write Theorem 3.1.9 with havens instead of (k,r)-linked sets, as done by Johnson et al. [24, 3.3], with the guarantee that the procedure runs in FPT time.

Theorem 3.1.11 (First main contribution)**.** *Let D be a digraph and k be a non-negative integer. There is an algorithm running in time $2^{\mathcal{O}(k\log k)} \cdot n^{\mathcal{O}(1)}$ that correctly states that D admits a haven of order k or produces an arboreal decomposition of D of width at most $3k-2$.*

Next, we discuss some of the steps in the proof of the Directed Grid Theorem.

3.1.1 Brambles and the Directed Grid Theorem

The Directed Grid Theorem is as stated below.

Theorem 3.1.12 (Kawarabayashi and Kreutzer [4])**.** *There is a function $f : \mathbb{N} \to \mathbb{N}$ such that given any directed graph and any fixed constant k, in polynomial time, we can obtain either*

1. *an arboreal decomposition of D of width at most $f(k)$, or*
2. *a cylindrical grid of order k as a butterfly minor of D.*

The proof of the Directed Grid Theorem [4] starts by asking if a digraph D has $\mathsf{dtw}(D) \leq f(k)$, for some integer k. By Theorem 3.1.11, an approximate answer to this question can be computed in FPT time with parameter $k \geq 0$. If a haven is obtained, the next step uses it to construct a bramble of large order. In order to justify our following results, we now discuss how to construct brambles from havens.

Finding a hitting set of minimum size of a bramble $\mathcal{B}$ is not an easy task. In general, in order to check whether a given set X is a hitting set of $\mathcal{B}$, the naive approach would be to go through all the elements of $\mathcal{B}$ and verify that X intersects each of them. Since a bramble $\mathcal{B}$ may contain $\Omega(2^n)$ elements, independently of its order, this procedure is not efficient. For instance, consider the digraph D shown in Fig. 13, which has vertex set $\{v_0, v_1, \ldots, v_n\}$ and edge set $\{(v_0, v_i) \cup (v_i, v_0) \mid i \in [n]\}$. The set $\mathcal{B} = \{D[X] \mid X \subseteq V(D)$ and $v_0 \in X\}$ is easily seen to be a bramble in D of order one and size $2^{|V(D)|-1}$ since there is an edge in D from every vertex in $V(D) \setminus \{v_0\}$ to v_0 and vice-versa. However, when $\mathcal{B}$ is the bramble obtained by a construction

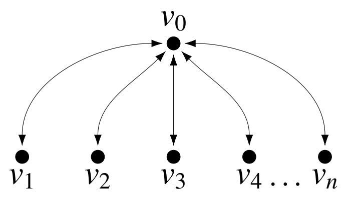


Figure 13 – Example of a digraph D having a bramble of order one and size $2^{|V(D)|-1}$. Here a bidirectional edge is used to represent a pair of edges in both directions.

used in a proof of Lemma 3.1.7, which we present below, then $|\mathcal{B}| = n^{\mathcal{O}(k)}$ and thus in this case we can find hitting sets of $\mathcal{B}$ of size k in XP time, and decide whether a given set $X \subseteq V(D)$ is a hitting set of $\mathcal{B}$ in XP time.

Lemma 3.1.7 implies that if D is a digraph admitting a haven of order $k+1$, then D contains a bramble of order at least $\lceil (k+1)/2 \rceil = \lfloor k/2 \rfloor + 1$. In fact, given such a haven, it is easy to construct the claimed bramble, as we proceed to explain. Namely, given a haven β of order $k+1$ in D, we define $\mathcal{B} = \{D[\beta(Z)] \mid Z \subseteq V(D)$ and $|Z| \leq \lfloor k/2 \rfloor\}$. Note that, since β is a haven, the elements of $\mathcal{B}$ are strongly connected subgraphs of D. We claim that any two elements of $\mathcal{B}$ intersect.

Indeed, let $B, B' \in \mathcal{B}$ and let $Z, Z' \subseteq V(D)$ such that $\beta(Z) = V(B)$ and $\beta(Z') = V(B')$. Since $|Z| \leq \lfloor k/2 \rfloor$ and $|Z'| \leq \lfloor k/2 \rfloor$, we have that $|Z \cup Z'| \leq k$, and since β is a haven of order $k+1$, it follows that $\beta(Z \cup Z') \subseteq \beta(Z) \cap \beta(Z') = V(B) \cap V(B')$ and therefore, in particular, $V(B) \cap V(B') \neq \emptyset$. Finally, let us argue about the order of $\mathcal{B}$. Consider an arbitrary vertex set $X \subseteq V(D)$ with $|X| \leq \lfloor k/2 \rfloor$. Since β is a haven or order $k+1 \geq \lfloor k/2 \rfloor$, there is a bramble element $\beta(X) \in \mathcal{B}$ with $V(\beta(X)) \cap X = \emptyset$, and thus $\mathsf{ord}(\mathcal{B}) \geq \lfloor k/2 \rfloor + 1$, as we wanted to prove. Moreover, since there is one element in $\mathcal{B}$ for each $Z \subseteq V(D)$ with $|Z| \leq \lfloor k/2 \rfloor$, we conclude that $|\mathcal{B}| = n^{\mathcal{O}(k)}$.

In [4], the authors show how to obtain, from a bramble $\mathcal{B}$ of order $k(k+2)$, a path P that is a hitting set of $\mathcal{B}$ containing a well-linked set A of size k.

Proposition 3.1.13 (Kawarabayashi and Kreutzer [4, Lemma 4.3 of the full version])**.** *Let D be a digraph and $\mathcal{B}$ be a bramble in D. Then there is a path P intersecting every $B \in \mathcal{B}$.*

Proposition 3.1.14 (Kawarabayashi and Kreutzer [4, Lemma 4.4 of the full version])**.** *Let D be a digraph, $\mathcal{B}$ be a bramble of order $k(k+2)$ in D, and $P = P(\mathcal{B})$ be a path intersecting every $B \in \mathcal{B}$. Then there is a set $A \subseteq V(P)$ of size k which is well-linked.*

Although the statements of the previous two propositions in [4] are not algorithmic, algorithms for both results can be extracted from the constructive proofs. However, the naive approach to decide if a set $X \subseteq V(D)$ is a hitting set of a bramble $\mathcal{B}$ is to check if $V(B) \cap X \neq \emptyset$ for each $B \in \mathcal{B}$. Thus the running time of the algorithms yielded by the proofs of Propositions 3.1.13 and 3.1.14 is influenced by the size of the bramble given as input. Although in general this is not efficient since, as discussed above, a bramble can have size $\Omega(2^n)$ even if it has small order, in the particular case where $\mathcal{B}$ is the bramble constructed from havens as presented above, those constructions yield XP algorithms with parameter k since $|\mathcal{B}| = n^{\mathcal{O}(k)}$.

In Section 3.3 we show that, when considering a particular choice of a bramble $\mathcal{B}$ which is constructed from (k,r)-linked sets, for appropriate choices of k and r, we can decide if a given set X is a hitting set of $\mathcal{B}$ in polynomial time and compute hitting sets of $\mathcal{B}$ in FPT time when parameterized by $\mathsf{ord}(\mathcal{B})$. Then, we show how to obtain a path P intersecting all elements of $\mathcal{B}$ in polynomial time, improving Proposition 3.1.13. We use this latter result to give an FPT algorithm with parameter $\mathsf{ord}(\mathcal{B})$ that produces, from a path P intersecting all elements of a bramble of large order, a well-linked set A of size k which is contained in $V(P)$.

Theorem 3.1.15 (Second main contribution)**.** *Let $g(k) = (k+1)(\lfloor k/2 \rfloor + 1) - 1$, D be a digraph and T be a $(g(k)-1, g(k)-1)$-linked set in D with $|T| = 2g(k)-1$. There is an algorithm running in time $2^{\mathcal{O}(k^2 \log k)} \cdot n^{\mathcal{O}(1)}$ that finds in D a bramble $\mathcal{B}$ of order $g(k)$, a path P that is a hitting set of $\mathcal{B}$, and a well-linked set A of order k such that $A \subseteq V(P)$.*

The request that we make on $\mathsf{ord}(\mathcal{B})$ is also an improvement when compared to Proposition 3.1.14. In the next section we give an overview of how a cylindrical grid is found in [4] from the output of Theorem 3.1.15. We discuss why the algorithms used in the remaining constructive steps of their proof are naturally FPT to obtain the following corollary.

Corollary 3.1.16. *Let k be a non-negative integer and D be a digraph. There is a function $f : \mathbb{N} \to \mathbb{N}$ and an FPT algorithm, with parameter k, that either*

1. *produces an arboreal decomposition of D of width at most $f(k)$, or*
2. *finds a cylindrical grid of order k as a butterfly minor of D.*

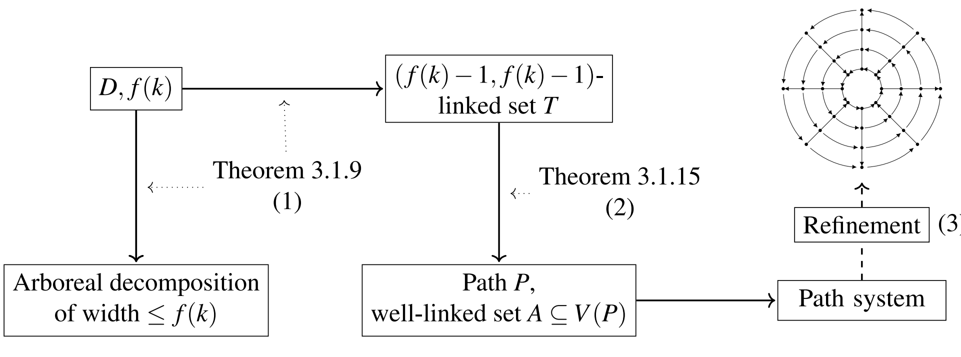
3.1.2 Finding a cylindrical grid

On a very high level, the proof of the Directed Grid Theorem [4] can be summarized in three parts. Using the terminology adopted in this paper, for a function f as in the statement of Theorem 3.1.12 and given a digraph D, we

(1) pipeline theorems 3.1.9 and 3.1.15 to either produce an arboreal decomposition of D of width at most $f(k)$ or construct P and A as in the statement of the latter;

(2) use P and A to construct a well-linked *path system* that is formed by a collection of paths; and

(3) iteratively refine the paths in the path system into new structures until a (butterfly) model of a cylindrical grid is obtained.



$D, f(k)$
$(f(k)-1, f(k)-1)$-linked set T
Theorem 3.1.9
(1)
Theorem 3.1.15
(2)
Refinement
(3)
Arboreal decomposition of width $\leq f(k)$
Path P, well-linked set $A \subseteq V(P)$
Path system

Figure 14 – Illustration of steps (1)-(3).

See Figure 14 for an illustration of those steps. As previously mentioned, we only improve on the procedures related to the first part and, in this section, we justify why this is sufficient to obtain Corollary 3.1.16. The main observations are that the algorithm runs maintaining and refining a collection of paths, where the size of the collection depends only on k, and that each of those refinements can be realized by iteratively testing how a given path intersects some subset of the collection. The number of tests depends only on k and each test is done in polynomial time. We show how to construct a path system from P and A, as mentioned in item (2) above. For our examples, it is convenient to adopt the following definitions from from the full version of [4].

Definition 3.1.17 (Linkages). *Let D be a digraph and $A, B \subseteq V(D)$ with $A \neq B$. A* linkage from A to B *in D, or an* (A,B)-Linkage*, is a set of of pairwise vertex-disjoint paths from A to B.*

Definition 3.1.18 (Path system). *Let D be a digraph and ℓ, p be two positive integers. An* ℓ-linked path system of order p *is a sequence $\mathcal{S}$ with $\mathcal{S} = (\mathcal{P}, \mathcal{L}, \mathcal{A})$ where*

- *$\mathcal{A} = \{A_i^{in}, A_i^{out} \mid i \in [p]\}$ where each A_i^{in} and each A_i^{out} is a well-linked set of order ℓ;*
- *$\mathcal{P}$ is a sequence $P_1, \ldots, P_p$ of pairwise vertex disjoint paths such that, for all $i \in [p]$, $V(P_i) \supseteq A_i^{in} \cup A_i^{out}$, every vertex in A_i^{in} appears in P_i before any vertex of A_i^{out}; and*
- *$\mathcal{L}$ is a collection $\{L_{i,j} \mid i, j \in [p]$ with $i \neq j\}$ of linkages where each $L_{i,j}$ is a linkage of size ℓ from A_i^{out} to A_j^{in}.*

Although the definition of path systems is quite loaded, it is not hard to visualize; see Figure 15 for an illustration. Notice that, knowing that the sets A^i_{in}, A^i_{out} are well-linked, a

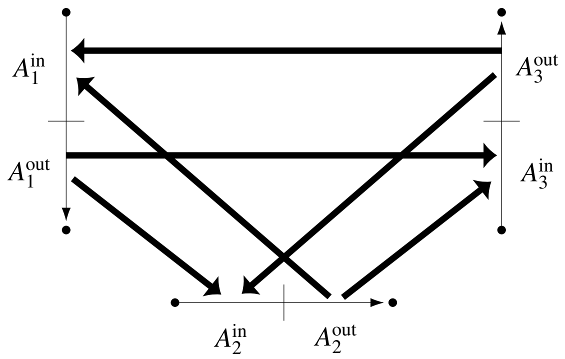


Figure 15 – An ℓ-linked path system of order $p(=3)$. A thick edge denotes a linkage of size ℓ from a set A_i^{out} to a set A_j^{in}, with $i \neq j$.

path system is entirely formed by paths behaving in a particular way: the collection $\mathcal{P}$ of size p, and the collection of paths appearing in the linkages $L_{i,j}$. Since each of those linkages has size ℓ, an ℓ-linked path system of order p is formed by $p+2\binom{p}{2}\ell$ paths. With this observation, the task of constructing a path system from the output of Theorem 3.1.15 becomes an easy one.

Assume that we are given a path P and a well-linked set A with $|A| = 2\ell \cdot p$ and $A \subseteq V(P)$. Let $\sigma = a_1, a_2, \ldots, a_{2\ell \cdot p}$ be an ordering of the vertices of A as they appear in P, from the first to the last vertex of the path. To construct an ℓ-linked path system of order p, we follow P in this order and, for $i \in [p]$, we define the path P_i to be the subpath of P from $a_{(i-1)2\ell+1}$ to $a_{i \cdot 2\ell}$. See Figure 16 for an illustration of this procedure. Since we know that A is well-linked, and clearly every subset of a well-linked set is also well-linked, we define A^i_{in} to be the set containing the first ℓ vertices of $V(P_i) \cap A$ and A^i_{out} to be last ℓ vertices of $V(P_i) \cap A$ with respect to σ.

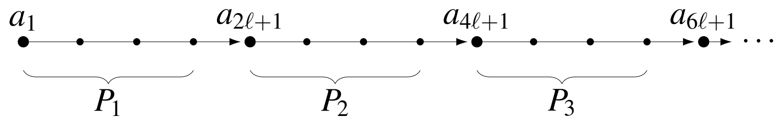
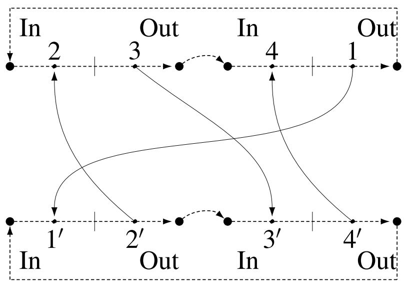


Figure 16 – Finding the paths P_i from P and A.

Next, for $i, j \in [p]$ with $i \neq j$, we choose $L_{i,j}$ to be a linkage from A^i_{out} to A^j_{in}. At least one choice for $L_{i,j}$ is guaranteed to exist because $A^i_{\text{in}} \cup A^j_{\text{out}} \subseteq A$ and A is well-linked. Moreover, we can find each linkage in polynomial time by applying Menger's Theorem (c.f. Theorem 2.0.1) and solving a flow problem. Hence given P and A of adequate size, we can find an ℓ-linked path system of order p in polynomial time.

As in Figure 17, it is easy to find a cylindrical grid in a sufficiently large path system if it is "well-behaved", that is, when every path in $\mathcal{L}$ is internally disjoint from all others. In fact, in such cases every P_i models one vertex of a *biclique* that is a butterfly minor of D. A biclique is a digraph H having a pair of edges in both directions between any two vertices of H. Clearly a biclique with $2k^2$ vertices contains a cylindrical grid of order k.



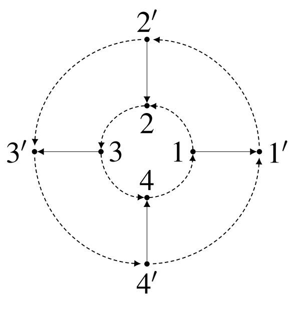


Figure 17 – An example of cylindrical grid of order 2 in a "well-behaved" path system, where we assume that every path in $\mathcal{L}$ is internally disjoint from all others.

Unfortunately, in general we cannot expect every path system to behave in this way. Hence the proof of the Directed Grid Theorem by Kawarabayashi and Kreutzer [4] follows a sequence of refinements, as mentioned in item (3) above, each constructing a new structure from the previous one until a cylindrical grid is obtained. This part is represented by the dashed edges in figures 9 and 14 and, although it is not hard to see that the algorithms realizing those constructions are naturally FPT, the constructive proofs are in fact the largest and most involved part of their paper. Namely, they show how to find a *web*[3] or a cylindrical grid from a path-system that is sufficiently large. If a web is obtained, then the next step is to find a *fence*[3] in it. Lastly, they prove that we are guaranteed to find a cylindrical grid of order k in any sufficiently large fence.

Fortunately, and as is the case with path systems, webs and fences are defined around collections of paths satisfying some properties that can be easilly verified in polynomial time. Since the number of paths in a ℓ-linked path system of order p depends on ℓ and p only, we can search for a web in a path system by testing the defining properties of webs for every subset of the set of paths in the path system. Thus in FPT time with parameters ℓ and p we can find a web in a path system. A similar approach is viable to find fences in webs and cylindrical grids in fences and thus Corollary 3.1.16 follows from Theorem 3.1.15.

3.2 Balanced separators and arboreal decompositions

The algorithm for arboreal decompositions given in [24] starts with a trivial decomposition $(\{r\}, \emptyset, \{W_r\})$ whose underlying arborescence contains only one vertex r. Thus, $W_r = V(G)$. Each iteration splits the vertices contained in an excessively large leaf of the current decomposition, if one exists, into a set of new leaves, while guaranteeing that the width of the non-leaf vertices remains bounded from above by a function of k. Although this problem is not explicitly named by the authors, on each of those split operations the algorithm has to decide whether the input digraph admits a (T,r)-balanced separator for a given set $T \subseteq V(D)$. Formally, on each iteration they need to solve a particular case of the following problem.

BALANCED SEPARATOR

Input: A digraph D, a set $T \subseteq V(D)$ of size k, and two non-negative integers r and s.

Output: A (T,r)-balanced separator Z with $|Z| \leq s$, if it exists.

[3] The definitions of *fences* can be found in the full version of [4].

The BALANCED SEPARATOR problem can be naively solved by checking all $\binom{n}{s}$ sets Z of size s in $V(D)$ and enumerating the strong components of $D \setminus Z$. Therefore it is in XP with parameter s. Furthermore, the process of finding balanced separators is the only step of the algorithm given in [24] that is done in XP time. In the next section, we show how to compute (T,r)-balanced separators in FPT time with parameter k. In particular, we show that a set Z is a (T,r)-balanced separator if and only if Z is a solution to a separation problem introduced in [56] that is a particular case of the MULTICUT problem in digraphs. Then, we use this result to improve the algorithm by Johnson et al. [24] for approximate arboreal decompositions (cf. Proposition 3.1.6), showing that it can be done in FPT time. Notice that we can assume that $r \leq k-1$ and $s \leq k-r-1$: if $r \geq k$, the empty set is a (T,r)-balanced separator and, if $s \geq k-r$, any choice of s vertices from T form a (T,r)-balanced separator. To avoid repetition, we make these considerations here and refrain from repeating them in the remainder of this article. We refer to instances of BALANCED SEPARATOR as (D,T,k,r,s)

3.2.1 *Computing (T,r)-balanced separators in FPT time*

Given a graph or digraph D and a set of pairs of terminal vertices $\{(s_1,t_1), (s_2,t_2), \ldots, (s_k,t_k)\}$, the MULTICUT problem asks to minimize the size of a set $Z \subseteq V(D)$ such that there is no path from s_i to t_i in $D \setminus Z$, for $i \in [k]$. When parameterized by the size of the solution, the problem is FPT in undirected graphs [10, 11]. On the directed case, this problem is FPT in DAGs when parameterized by the size of the solution and the number of pairs of terminals [59], but W[1]-hard in the general case even for a request set of size 4 [13].

A variation of MULTICUT is considered in [56]. Namely, in the LINEAR EDGE CUT problem, we are given a digraph D and a collection of sets of vertices $\{S_1, \ldots, S_k\}$, and we want to find a minimum set of edges Z such that there is no path from S_i to S_j in $D \setminus Z$ whenever $j > i$. We remark that the authors in [56] refer to this problem as LINEAR CUT only. This problem is FPT when parameterized by the size of the solution:

Proposition 3.2.1 (Erbacher et al. [56])**.** *The* LINEAR EDGE CUT *problem can be solved in time $\mathcal{O}(4^s \cdot s \cdot n^4)$, where s is the size of the solution.*

We remark that the authors of [56] mention that this result can also be achieved by using a reduction to the SKEW SEPARATOR algorithm given in [60].

In this section, we show how to use the algorithm for the LINEAR EDGE CUT problem to solve the vertex version, and then show how this version can be used to compute (T,r)-balanced separators in FPT time. We formally define the vertex version below.

LINEAR VERTEX CUT

Input: A digraph D, a collection of terminal sets $\mathcal{T}$, with $\mathcal{T} = \{T_1, T_2, \dots, T_k\}$, where $T_i \subseteq V(D)$ for $i \in [k]$, and an integer $s \geq 0$.

Output: A set of vertices $Z \subseteq V(D)$ with $|Z| \leq s$ such that there are no paths in $D \setminus Z$ from T_i to T_j, for $1 \leq i < j \leq k$.

From an instance $(D, \mathcal{T}, s)$ of LINEAR VERTEX CUT, we construct an equivalent instance of $(D', \mathcal{T}', s)$ of LINEAR EDGE CUT as follows. First, notice that any vertex v occurring in the intersection of two distinct sets in $\mathcal{T}$ must be part of any solution for the instance. Thus we can assume that every vertex of D occurs in at most one set in $\mathcal{T}$. Now, for each vertex $v \in V(D)$, add to D' two vertices v_{in} and v_{out} and an edge e_v from v_{in} to v_{out}. For each edge $e \in E(D)$ with tail u and head v, add to D' a set of $s+1$ parallel edges from u_{out} to v_{in}. Finally, for each $v \in T_i$, for $i \in [k]$, add a new vertex v' to D' together with $s+1$ edges from v' to v_{in} and $s+1$ edges from v_{out} to v'. Let $T_i' = \{v' \mid v \in T_i\}$ and $\mathcal{T}' = \{T_1', \dots, T_k'\}$. We have the following easy lemma.

Lemma 3.2.2. *An instance $(D, \mathcal{T}, s)$ of* LINEAR VERTEX CUT *is positive if and only if the associated instance $(D', \mathcal{T}', s)$ of* LINEAR EDGE CUT *is positive.*

Proof. Let $Z \subseteq V(D)$ be a solution for $(D, \mathcal{T}, s)$ and $Z' = \{e_v \mid v \in Z\} \subseteq E(D')$. By contradiction, assume that there is a path P' in $D' \setminus Z'$ from a vertex u' to a vertex v', for $u' \in T_i'$, $v' \in T_j'$, and $j > i$. Then there is a path P from u to v in $D \setminus Z$ with vertex set $\{v \mid e_v \in E(P')\}$. This contradicts our choice of Z and thus the necessity holds.

For the sufficiency, let Z' be a minimal solution for $(D', \mathcal{T}', s)$. Notice that all edges in Z' are from a vertex v_{in} to its respective v_{out}, as the budget s for the size of Z' does not allow any other choice. Let $Z = \{v \mid e_v \in Z'\}$ and, by contradiction, let P be a path in $D \setminus Z$ from a vertex u to a vertex v, with $u \in T_i$, $v \in T_j$, and $j > i$. For each edge $e \in E(P)$ with $e = (x,y)$ there is an edge e' with $e' = (x_{\mathsf{out}}, y_{\mathsf{in}})$ in $D' \setminus Z'$. Let F' be the set of such edges of D'. Now, there is a path P' from u_{in} to v_{out} in D' with edge set $\{e_v \mid v \in V(P)\} \cup F'$. Appending to P' the edges from u' to u_{in} and from v_{out} to v' we construct a path from u' to v' in $D \setminus Z'$, contradicting our choice of Z'. Therefore, the sufficiency also holds and the lemma follows. $\square$

Combining Proposition 3.2.1 and Lemma 3.2.2 we get the following.

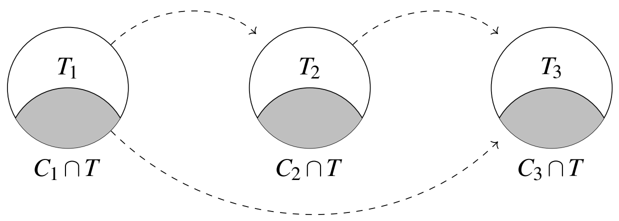


Figure 18 – Illustration of the construction of the sets T_i. In the figure, we include only the strongly connected components of $D \setminus Z$ intersecting T. Dashed edges indicate that there are no paths from C_i to C_j whenever $i > j$ (and thus no path from T_i to T_j whenever $i > j$).

Corollary 3.2.3. *There is an FPT algorithm for the* LINEAR VERTEX CUT *problem parameterized by the size s of the solution and running in time $\mathcal{O}(4^s \cdot s \cdot n^4)$.*

We now show how to solve BALANCED SEPARATOR using LINEAR VERTEX CUT. Namely, we show that a digraph D admits a (T,r)-balanced separator Z if and only if Z is a solution to some instance $(D, \mathcal{T}, s)$ of LINEAR VERTEX CUT where $\mathcal{T}$ depends of T.

Lemma 3.2.4. *Let (D,T,k,r,s) be an instance of* BALANCED SEPARATOR. *A set $Z \subseteq V(D)$ with $|Z| \leq s$ is a (T,r)-balanced separator if and only if there is a partition $\mathcal{T}$ of T into sets $T_1, T_2, \dots, T_\ell$ such that*

1. *$|T_i| \leq r$, for $i \in [\ell]$, and*
2. *Z is a solution for the instance $(D, \mathcal{T}, s)$ of* LINEAR VERTEX CUT.

Proof. For the necessity, let Z be a (T,r)-balanced separator with $|Z| \leq s$. Let $\mathcal{C}$ be the set of strong components of $D \setminus Z$ and consider an ordering $C_1, \dots, C_\ell$ of its elements such that there is no path from C_i to C_j in $D \setminus Z$ whenever $j > i$. Notice that this is the reverse of a topological ordering for the elements of $\mathcal{C}$. Let $v_1, \dots, v_q$ be the vertices in $T \cap Z$, if any exist. For $i \in [\ell]$, choose $T_i = V(C_i) \cap T$ and define $\mathcal{T} = \{T_1, T_2, \dots, T_\ell\}$ if $T \cap Z \neq \emptyset$ or $\mathcal{T} = \{T_1, T_2, \dots, T_\ell, \{v_1\}, \dots, \{v_q\}\}$ otherwise. See Figure 18 for an illustration of this construction. Notice that it is possible for a set T_i to be empty.

Since Z is a (T,r)-balanced separator, we know that $|T_i| \leq r$ holds for all $i \in [\ell]$. Since the vertices in a non-empty set T_i are contained in exactly one strong component of $D \setminus Z$, any path between different sets in $\mathcal{T}$ must contain a path between distinct strong components of $D \setminus Z$. Thus we conclude that there are no paths from a set T_i to another set T_j with $j > i$, since otherwise we would have a contradiction to our choice for the order of the elements of $\mathcal{C}$, and therefore Z is a solution for the instance $(D, \mathcal{T}, s)$ of LINEAR VERTEX CUT.

For the sufficiency, let $\mathcal{T}$ be as in the statement of the lemma and Z be a solution for the instance $(D, \mathcal{T}, s)$ of LINEAR VERTEX CUT. First, notice that no strong component of $D \setminus Z$ can intersect two distinct sets $T, T' \in \mathcal{T}$. Indeed, if this were the case, then there would be a path in $D \setminus Z$ from a vertex in T to a vertex in T' and vice-versa, contradicting the fact that Z is a solution for $(D, \mathcal{T}, s)$. Thus, if $|V(C) \cap T| \geq r+1$ for some strong component C of $D \setminus Z$, we have a contradiction as C would intersect at least two distinct sets in $\mathcal{T}$. We conclude that Z is a (T, r)-balanced separator and the lemma follows. □

The FPT algorithm for BALANCED SEPARATOR follows from Lemma 3.2.4 and Corollary 3.2.3. The running time is heavily tied to the number of partitions $\mathcal{T}$ that can be generated from a given set T of an instance (D, T, k, r, s) of BALANCED SEPARATOR. This value is bounded by the k-th *ordered Bell number* [61]. The *Bell number* [62] counts the number of partitions of a set, and its ordered variant also considers the number of possible orderings for each partition. The k-th ordered Bell number is of the form $2^{\mathcal{O}(k \log k)}$. From the previous discussion we get the following theorem.

Theorem 3.2.5. *There is an algorithm running in time $\mathcal{O}(4^k \cdot 2^{\mathcal{O}(k \log k)} \cdot k \cdot n^{\mathcal{O}(1)})$ for the* BALANCED SEPARATOR *problem.*

Proof. Let (D, T, k, r, s) be an instance of BALANCED SEPARATOR and $\mathcal{T}^*$ be the set of all ordered partitions $\{T_1, \ldots, T_\ell\}$ of T with $|T_i| \leq r$, for $i \in [\ell]$.

By Corollary 3.2.3, we can solve instances of LINEAR VERTEX CUT problem in time $\mathcal{O}(4^s \cdot s \cdot n^4)$ for s being the size of the solution. By Lemma 3.2.4, Z is a (T, r)-balanced separator if and only if there is a $\mathcal{T} \in \mathcal{T}^*$ such that the instance $(D, \mathcal{T}, s)$ of LINEAR VERTEX CUT is positive. Finally, since $|\mathcal{T}^*|$ is at most the k-th ordered Bell number, we can solve BALANCED SEPARATOR by testing $2^{\mathcal{O}(k \log k)}$ instances of LINEAR VERTEX CUT. As $s \leq k - r$ (since otherwise the instance of BALANCED SEPARATOR is trivially positive), the bound on the running time follows. □

3.2.2 An FPT algorithm for approximate arboreal decompositions

We are now ready to prove Theorem 3.1.9. We remark that the proof below follows [24, 3.3] except that we replace the XP procedure of the proof by our FPT algorithm for k-BALANCED SEPARATOR. In the following proof, we need to test whether a given set $T \subseteq V(D)$ admits a $(T, k-1)$-balanced separator of size at most $k-1$. Thus we remind the reader of the discussion made in the beginning of Section 3.2: if $|T| \leq 2k-2$, then the answer is positive since we can pick any $k-1$ vertices of T to form a solution.

Theorem 3.1.9. *Let D be a digraph and k be a non-negative integer. There is an algorithm running in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ that either produces a nice arboreal decomposition of D of width at most $3k-2$ or outputs a $(k-1,k-1)$-linked set T with $T = 2k-1$.*

Proof. We begin with a nice arboreal decomposition $(R_0, \mathcal{X}_0, \mathcal{W}_0)$ of D where $\mathcal{X}_0 = \emptyset$, $V(R_0) = \{r\}$, and $\mathcal{W}_0 = V(D)$. We maintain an arboreal decomposition $(R, \mathcal{X}, \mathcal{W})$ of D for which the following two properties hold:

(P1) $|W_r \cup (\bigcup_{e \sim r} X_e)| \leq 3k-1$ for every $r \in V(R)$ of out-degree at least one, and

(P2) $|X_e| \leq 2k-1$ for every $e \in E(R)$.

Notice that both (P1) and (P2) hold for $(R_0, \mathcal{X}_0, \mathcal{W}_0)$.

If (P1) holds for all $r \in V(R)$, then we have constructed an arboreal decomposition with the desired width. Otherwise, we can assume that $(R, \mathcal{X}, \mathcal{W})$ contains at least one leaf that is *too large*. That is, the width of a vertex r_0 of out-degree zero of R is at least $3k$. If there is an edge $e_0 \in E(R)$ with head r_0, let $T = X_{e_0}$. Otherwise, let $T = \emptyset$. Either way, $|T| \leq 2k-1$ and $|W_{r_0}| \geq 3k - |T| \geq k+1$.

Now, we test whether D contains a $(T, k-1)$-balanced separator of size at most $k-1$ and, by Theorem 3.2.5, this test can be done in time $\mathcal{O}(4^k \cdot 2^{\mathcal{O}(k \log k)} \cdot k \cdot n^{\mathcal{O}(1)})$. If $|T| \leq 2k-2$ then the answer is positive since we can pick any set of $k-1$ vertices of T to form a solution. Thus, if the answer is negative, we have $|T| = 2k-1$ and we terminate the algorithm outputting T. We may now assume that D contains a $(T, k-1)$-balanced separator Z' with $|Z'| \leq k-1$.

From the bound on the sizes of the sets, there are at least two vertices in $W_{r_0} \setminus Z'$. Choose v to be any of those two vertices, and let $Z = Z' \cup \{v\}$. Now $|Z| \leq k$, $Z \cap W_{r_0} \neq \emptyset$, and $|V(C) \cap T| \leq k-1$ holds for every strong component C of $D \setminus Z$.

Let $C_1, \ldots, C_\ell$ be the strong components of $D \setminus Z$. If B is a strong component of $C_i \setminus T$, for $i \in [\ell]$, then either $V(B) \subseteq W_{r_0}$ or $V(B) \cap W_{r_0} = \emptyset$, for W_{r_0} is T-guarded. Let $B_1, \ldots, B_d$ be all such strong components for which $V(B_j) \subseteq W_{r_0}$ for all $j \in [d]$. Furthermore, let $f : \mathbb{N} \to \mathbb{N}$ be a function assigning an index j to an index i if and only if $B_i \subseteq C_j \setminus T$. Thus, f can be used to tell which set C_i contains a given B_j. Now, $Z \cap W_{r_0}, V(B_1), \ldots, V(B_d)$ is a partition of W_{r_0} into non-empty sets. We show that this partition yields another arboreal decomposition of D.

Let R' be the arborescence obtained from R by adding a vertex r_i and an edge e_i from r_0 to r_i, for $i \in [d]$. Furthermore, let $X'_e = X_e$ for all $e \in E(R)$ and $W'_r = W_r$ for all $r \in V(R) \setminus \{r_0\}$.

Also, let $W'_{r_0} = W_{r_0} \cap Z$ and, for $i \in [d]$, let $X'_{e_i} = Z \cup (V(C_{f(i)}) \cap T)$ and $W'_{r_i} = V(B_i)$. Finally, define $\mathcal{X}' = \{X'_e \mid e \in E(R')\}$ and $\mathcal{W}' = \{W'_r \mid r \in V(R')\}$. As the vertices of W_{r_0} have been spread into non-empty sets, we only need to verify that $(R', \mathcal{X}', \mathcal{W}')$ is an arboreal decomposition of D for which (P1) and (P2) hold; see Figure 19 for an illustration.

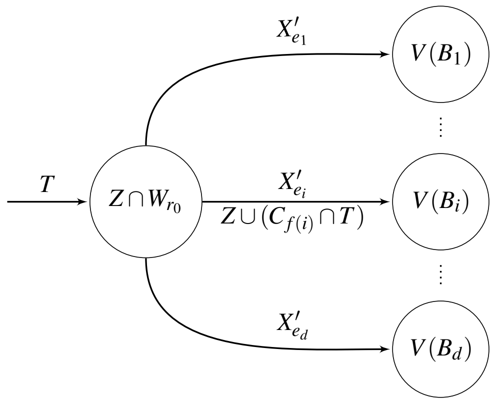


Figure 19 – Spreading the vertices in W_{r_0}.

$\mathcal{W}'$ is indeed a partition of $V(D)$ into non-empty sets, as W_{r_0} is partitioned into non-empty sets. For $i \in [d]$, $W'_{r_i} = V(B_i)$ and B_i is a strong component of $C_{f(i)} \setminus T$. Thus, each new leaf r_i added to R is such that W'_{r_i} is X'_{e_i}-guarded and, for all $e \in E(R')$, $\bigcup\{W'_r : r \in V(R'), r > e\}$ is X'_e-guarded as the property remains unchanged for all $e \in E(R)$.

For $r \in V(R)$, the validity of (P1) remains unchanged. The width of r_0 is bounded from above by $|T| + |Z| \leq 2k - 1 + k = 3k - 1$, as desired, for $W'_{r_0} \subseteq Z$ and $\bigcup_{e \sim r_0} X'_e \subseteq T \cup Z$. (P2) remains true in $(R', \mathcal{X}', \mathcal{W}')$ for all $e \in E(R)$. For e_i, $i \in [d]$, $|X_{e_i}| \leq |Z| + |V(C_{f(i)}) \cap T|$. By the assumption that $(D, T, 2k-1, k-1, k-1)$ is a positive instance of BALANCED SEPARATOR, $|Z| + |V(C_{f(i)} \cap T| \leq k + k - 1 = 2k - 1$.

Observe that, since each B_i is disjoint from $T \cup Z$, $(R', \mathcal{X}', \mathcal{W}')$ is actually a *nice* arboreal decomposition.

Now, if no leaf of $(R', \mathcal{X}', \mathcal{W}')$ is too large, we end the algorithm returning this arboreal decomposition of D. Otherwise, we repeat the aforementioned procedure with new choices for T and W_{r_0}.

Finally, the running time holds by Theorem 3.2.5, since $\mathcal{W}$ partitions $V(D)$ into non-empty sets and each iteration decreases the number of vertices in leaves that have width at least $3k$. $\square$

The proof of Theorem 3.1.11 easily follows from Lemma 3.1.10 and Theorem 3.1.9.

Theorem 3.1.11 (First main contribution)**.** *Let D be a digraph and k be a non-negative integer. There is an algorithm running in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ that correctly states that D admits a haven of order k or produces an arboreal decomposition of D of width at most $3k-2$.*

Proof. Applying Theorem 3.1.9 with input D, we either produce an arboreal decomposition of D of width at most $3k-2$ or find a set $T \subseteq V(D)$ with $|T| = 2k-1$ such that there is no $(T, k-1)$-balanced separator in D. Now, by Lemma 3.1.10 applied with inputs D, T, $r = k-1$, and $s = k-1$, we conclude that D admits a haven of order k and the result follows. □

Next, we show to use Theorem 3.1.9 to construct a bramble in digraphs of large directed tree-width that is easier to work with than the usual construction that depends on havens (see, for instance, [58, Chapter 6]).

3.3 Brambles and well-linked systems of paths

Let T be the set constructed by Theorem 3.1.9 applied to a digraph D with n vertices and $\mathsf{dtw}(D) \geq 3k-1$ and $\mathcal{H}$ be the haven obtained by applying Lemma 3.1.10 with input D and T. We remark that from $\mathcal{H}$ it is possible to construct a bramble $\mathcal{B}$ of order $\lfloor k/2 \rfloor$ and size $|V(D)|^{\mathcal{O}(k)}$ (see the discussion in Section 3.1.1). In this particular case the naive approach yields an XP algorithm to find a hitting set of $\mathcal{B}$ of size k in XP time with parameter k, by checking all $\binom{n}{k}$ subsets X of $V(D)$ with size k and testing whether $X \cap V(B) \neq \emptyset$ for each $B \in \mathcal{B}$, and thus XP algorithms can be extracted from the constructive proofs of Propositions 3.1.13 and 3.1.14 assuming that these properties hold for the input brambles. In Section 3.3.1, we show how to construct from T a bramble $\mathcal{B}_T$ of order k in digraphs with directed tree-width at least $3k-1$ that skips havens and is more efficient in the following two ways.

First, this construction allows us to verify if an induced subgraph D' of D contains an element of $\mathcal{B}_T$ by looking only at the strong components of D'. This allows us to test if a given set $X \subseteq V(D)$ is a hitting set of $\mathcal{B}_T$ in polynomial time. Second, we show that a set $Y \subseteq V(D)$ is a minimum hitting set of $\mathcal{B}_T$ if and only if Y is a solution for an appropriately defined instance of BALANCED SEPARATOR. Since we showed that this problem is FPT with parameter $|T|$ (Theorem 3.2.5), we can compute hitting sets of $\mathcal{B}_T$ in FPT time with parameter $\mathsf{ord}(\mathcal{B}_T)$. Then, in Section 3.4 we use those results to prove stronger versions of Propositions 3.1.13 and 3.1.14.

3.3.1 Brambles in digraphs of large directed tree-width

We now define T*-brambles* and some of its properties when T is the set obtained by applying Theorem 3.1.9 to a digraph D with $\mathsf{dtw}(D) \geq 3k-1$.

Definition 3.3.1. *Let D be a digraph and $T \subseteq V(D)$ with $|T| = 2k-1$. The T-*bramble *$\mathcal{B}_T$ of D is defined as*

$$\mathcal{B}_T = \{B \subseteq D \mid B \textit{ is induced, strongly connected, and } |V(B) \cap T| \geq k\}.$$

Notice that $\mathcal{B}_T$ is a bramble since, as $|T| = 2k-1$, any two of its element intersect. We remark that, in general, it is possible that $\mathsf{ord}(\mathcal{B}_T)$ is very small: it is in fact zero if, for example, no two vertices of T lay in the same strong component of D. Note also that $\mathcal{B}_T$ may be empty if, for instance, any strong component of D has size strictly smaller than k.

Lemma 3.3.2. *Let D be a digraph and T be a $(k-1,k-1)$-linked set of size $2k-1$ in D. Then the T-bramble $\mathcal{B}_T$ is a bramble of order k and a set $X \subseteq V(D)$ is a hitting set of $\mathcal{B}_T$ if and only if X is a $(T,k-1)$-balanced separator.*

Proof. Let D, T and $\mathcal{B}_T$ be as in the statement of the lemma. Since $|T| = 2k-1$, any set containing k vertices of T is a hitting set of $\mathcal{B}$. Thus $\mathsf{ord}(\mathcal{B}_T) \leq k$. Let $Z \subseteq V(D)$ with $|Z| \leq k-1$. By definition of $(k-1,k-1)$-linked sets, D does not contain any $(T,k-1)$-balanced separator of size $k-1$, and hence there is no strong component B of $D \setminus Z$ such that $|V(B) \cap T| \geq k$. Since $V(B) \cap Z = \emptyset$ and $B \in \mathcal{B}_T$, we conclude that Z is not a hitting set of $\mathcal{B}_T$ and therefore $\mathsf{ord}(\mathcal{B}_T) = k$.

For the second part of the lemma, let X be a hitting set of $\mathcal{B}_T$. Then $|V(C) \cap T| \leq k-1$ holds for every strong component C of $D \setminus X$ and, by definition, X is a $(T,k-1)$-balanced separator. Similarly, if X is a $(T,k-1)$-balanced separator then, by definition of $\mathcal{B}_T$, X is a hitting set of $\mathcal{B}_T$ and the result follows. □

Note that we can check whether a given set $X \subseteq V(D)$ is a hitting set of $\mathcal{B}_T$ by enumerating the strong components of $D \setminus X$ and, for each such a component C, checking whether $|V(C) \cap T| \geq k$. This can be done in time $\mathcal{O}(n+m)$. For the remaining of this section, and unless stated otherwise, let T be a $(k-1,k-1)$-linked set with $|T| = 2k-1$. In what follows, we use T-brambles to adapt Proposition 3.1.14 into an FPT algorithm.

To prove our version of Proposition 3.1.14, we start with a T-bramble $\mathcal{B}_T$ of order $g(k)$ (the value of $g(k)$ is specified later) in a digraph D with $\mathsf{dtw}(D) \geq 3g(k)-1$, and then we show how to find in polynomial time a path $P(\mathcal{B}_T)$ that is a hitting set of $\mathcal{B}_T$, adapting the proof of Proposition 3.1.13 shown in [4, Lemma 4.3 of the full version]. Next, we need to show how to split $\mathcal{B}_T$ into brambles of order at least $\lceil k/2 \rceil$ whose elements are intersected by subpaths of $P(\mathcal{B}_T)$. We do this by growing a subpath of P' of $P(\mathcal{B}_T)$ iteratively while checking, on each iteration, whether the set $\mathcal{B}'_T$ of elements of $\mathcal{B}_T$ intersecting $V(P')$ is a bramble of adequate order.

We now show how our choice of $\mathcal{B}_T$ allows us to estimate the order of $\mathcal{B}'_T$ by computing the order of its "complement bramble" $\mathcal{B}_T \setminus \mathcal{B}'_T$, and we show how to do this procedure in FPT time with parameter $\mathsf{ord}(\mathcal{B}_T)$. These ideas are formalized by the following definitions and results.

Definition 3.3.3. *Let $X \subseteq V(D)$ and $\mathcal{B}$ be a bramble in D. The* restricted bramble $\mathcal{B}(X)$ *contains the elements of $\mathcal{B}$ intersecting X and its* complement bramble $\overline{\mathcal{B}}(X)$ *contains the elements of $\mathcal{B}$ disjoint from X. Formally,*

$$\mathcal{B}(X) = \{B \in \mathcal{B} \mid V(B) \cap X \neq \emptyset\},$$

$$\overline{\mathcal{B}}(X) = \{B \in \mathcal{B} \mid V(B) \cap X = \emptyset\}.$$

Notice that both $\mathcal{B}(X)$ and $\overline{\mathcal{B}}(X)$ are brambles, as both are subsets of a bramble $\mathcal{B}$. Additionally, $\mathcal{B}(X)$ is disjoint from $\overline{\mathcal{B}}(X)$ and the union of a hitting set of the former with a hitting set of the latter is a hitting set of $\mathcal{B}$. From this remark, we have that

$$\mathsf{ord}(\mathcal{B}(X)) + \mathsf{ord}(\overline{\mathcal{B}}(X)) \geq \mathsf{ord}(\mathcal{B}), \tag{3.1}$$

and although in general the order of $\mathcal{B}(X)$ is hard to compute, we can estimate it by knowing the order of its complement bramble $\overline{\mathcal{B}}(X)$ and $\mathsf{ord}(\mathcal{B})$.

Consider now the brambles $\mathcal{B}_T$, $\mathcal{B}_T(X)$, and $\overline{\mathcal{B}_T}(X)$ for some $X \subseteq V(D)$. The following results show that hitting sets of $\overline{\mathcal{B}_T}(X)$ are exactly $(T \setminus X, k-1)$-balanced separators in $D \setminus X$.

Lemma 3.3.4. *Let $X, Z \subseteq V(D)$ and B be a strongly connected subgraph of D. Then $B \in \overline{\mathcal{B}_T}(X)$ and $V(B) \cap Z = \emptyset$ if and only if B is a strongly connected subgraph of $D \setminus (Z \cup X)$ with $|V(B) \cap T| \geq k$.*

Proof. For the necessity, assume that $B \in \overline{\mathcal{B}_T}$ and $V(B) \cap Z = \emptyset$. Then by the definition of $\overline{\mathcal{B}_T}(X)$, B is a strongly connected subgraph of $D \setminus (Z \cup X)$ intersecting T in at least k vertices.

For the sufficiency, assume that B is a strongly connected subgraph of $D \setminus (Z \cup X)$ containing at least k vertices of T. Then $B \in \overline{\mathcal{B}_T}(X)$ by the definition of $\overline{\mathcal{B}_T}(X)$ and the lemma follows since it is disjoint from $Z \cup X$. $\square$

The contrapositive of Lemma 3.3.4 characterizes hitting sets of $\overline{\mathcal{B}_T}(X)$.

Corollary 3.3.5. *Let $X, Z \subseteq V(D)$. Z is a hitting set of $\overline{\mathcal{B}_T}(X)$ if and only if Z is a $(T \setminus X, k-1)$-balanced separator in $D \setminus X$.*

Therefore, we can decide whether $\mathsf{ord}(\overline{\mathcal{B}_T}(X)) \leq s$ by testing whether D admits a $(T \setminus X, k-1)$-balanced separator of size s. The following result is a direct consequence of Theorem 3.2.5 and Corollary 3.3.5.

Corollary 3.3.6. *For any $X \subseteq V(D)$, there is an algorithm running in time $\mathcal{O}(4^k \cdot 2^{\mathcal{O}(k \log k)} \cdot k \cdot n^{\mathcal{O}(1)})$ that decides whether $\mathsf{ord}(\overline{\mathcal{B}_T}(X)) \leq s$.*

Next, we show how to find such a path $P(\mathcal{B}_T)$ as described above and a well-linked set A of size roughly $\sqrt{2k}$ that is contained in $V(P(\mathcal{B}_T))$.

3.4 Finding $P(\mathcal{B}_T)$ and A

The next lemma is an adaptation of the proof of [4, Lemma 4.3 of the full version] to our scenario. We exploit the fact that we can check whether a given set of vertices is a hitting set of $\mathcal{B}_T$ in polynomial time: by Lemma 3.3.2, a set $X \subseteq V(D)$ is a hitting set of $\mathcal{B}_T$ if and only if X is a $(T, k-1)$-balanced separator, and we can check if a given set X is a $(T, k-1)$-balanced separator by enumerating the strong components of the input digraph.

Lemma 3.4.1. *Let D be a digraph, T be a $(k-1, k-1)$-linked set of size $2k-1$, and consider the T-bramble $\mathcal{B}_T$. There is an algorithm running in time $\mathcal{O}(n(n+m))$ that produces a path P that is a hitting set of $\mathcal{B}_T$.*

Proof. If $\mathsf{ord}(\mathcal{B}_T) \geq 1$, then there is an element $B \in \mathcal{B}$ and a strong component C of D such that $V(B) \subseteq C$ and, by the definition of $\mathcal{B}$, we know that $D[C] \in \mathcal{B}_T$. Define $B_1 = D[V(C)]$, let v_1 be any vertex of B_1, and define P_1 as the path containing only the vertex v_1 and $V(P_0) = \emptyset$. We proceed to grow a path by iterating from P_1 to $P_{k'}$ where they all start from v_1, each P_i with $i \geq 2$ contains P_{i-1}, and $P_{k'}$ is a hitting set of $\mathcal{B}_T$. Throughout our process, we maintain a collection of elements $B_i \in \mathcal{B}_T$ such that $V(P_i)$ intersects $V(B_i)$ only in the last vertex v_i of P_i.

Since $|V(P_1)| = 1$ and $v_1 \in T \subseteq V(B_1)$, this condition trivially holds for P_1. Assume now that i paths have been chosen this way, with $i \geq 1$.

Consider the last vertex v_i of the path P_i and the element B_i of $\mathcal{B}_T$ with $V(P_i) \cap V(B_i) = \{v_i\}$. By Lemma 3.3.2, $V(P_i)$ is a hitting set of $\mathcal{B}_T$ if and only if $V(P_i)$ is a $(T, k-1)$-balanced separator, and this can be tested in time $\mathcal{O}(n+m)$ by enumerating all strong components of $D \setminus V(P_i)$. If $V(P_i)$ is a hitting set of $\mathcal{B}_T$, then we terminate the algorithm returning P_i. Otherwise, $V(P_i)$ is not a $(T, k-1)$-balanced separator and thus there is a strong component C of $D \setminus V(P_i)$ with $|V(C) \cap T| \geq k$. Therefore, $D[V(C)]$ is an element of $\mathcal{B}_T$ whose vertices are disjoint from $V(P_i)$ and we choose $B_{i+1} = D[V(C)]$.

Since $\mathcal{B}_T$ is a bramble, we can find a path P' from $v_i \in V(P_i) \cap V(B_i)$ to a vertex $v_{i+1} \in B_{i+1}$ in $D[V(B_i) \cup V(B_{i+1})]$ such that $V(P') \cap V(B_{i+1}) = \{v_{i+1}\}$. Moreover, v_i is the only vertex of P_i in B_i and thus the path P' does not contain any vertex in $V(P_i) \setminus \{v_i\}$. Now, let P_{i+1} be the path obtained from P_i by appending P'. By our choice of P', we know that only the last vertex v_{i+1} of P_{i+1} is in $V(B_{i+1})$, as desired, and $V(P_{i+1})$ hits strictly more elements of $\mathcal{B}_T$ than $V(P_i)$. We repeat the aforementioned procedure now considering the vertex v_{i+1}, the path P_{i+1}, and the element B_{i+1} of $\mathcal{B}_T$.

Since we can enumerate the strong components of a subgraph of D in time $\mathcal{O}(n+m)$ (see, for instance, [30, Chapter 6]), at the i-th iteration we can find B_{i+1}, the path P_{i+1}, and the vertex v_{i+1} in time $\mathcal{O}(n+m)$. Finally, the procedure eventually terminates as $|V(P)| \leq n$ and thus the bound on the running time follows. $\square$

For the remainder of this section, we assume that $g(k) = (k+1)(\lfloor k/2 \rfloor + 1) - 1$, that D is a digraph containing a $(g(k)-1, g(k)-1)$-linked set T of size $2g(k)-1$, consider the T-bramble $\mathcal{B}_T$ and fix P to be the path received by applying Lemma 3.4.1 with inputs D, T, and $\mathcal{B}_T$. To prove Theorem 3.1.16, we use the following definition.

Definition 3.4.2 ((i)-splits)**.** *An* (i)-split $\mathcal{S}$ *of* P *is a collection formed by a set* $\{Q_j \mid j \in [i]\}$ *of subpaths of* P, *a subpath* P_i *of* P, *a set of brambles* $\{\mathcal{B}_j \mid j \in [i]\}$, *a set of vertices* $\{a_j \mid j \in [i]\}$, *and a set of vertices* X_i *such that*

1. *for* $j \in [i]$, *vertex* a_j *is the sucessor in* P *of the last vertex of* Q_j, *and, if* $j \leq i-1$, *the first vertex of* Q_{j+1} *is the sucessor in* P *of vertex* a_j,
2. *for* $j \in [i]$, $\mathit{ord}(\mathcal{B}_j) \geq \lfloor k/2 \rfloor$,
3. *for* $j \in [i]$, $\mathcal{B}_j \subseteq \mathcal{B}_T$ *and* $V(Q_j)$ *is a hitting set of* $\mathcal{B}_j$,

4. *P_i is the subpath of P from the sucessor in P of the last vertex of Q_i to the last vertex of P, and*
5. *$X_i = \bigcup_{j\in[i]}(V(P_j)\cup\{a_j\})$, and*

$$\mathit{ord}(\overline{\mathcal{B}}(X_i)) \geq g(k) - i\left(\left\lfloor\frac{k}{2}\right\rfloor+1\right).$$

See Figure 20 for an example of a (2)-split. We remark that a (0)-split for P consists only of the path P_0 with $P_0 = P$ and the empty set X_0.

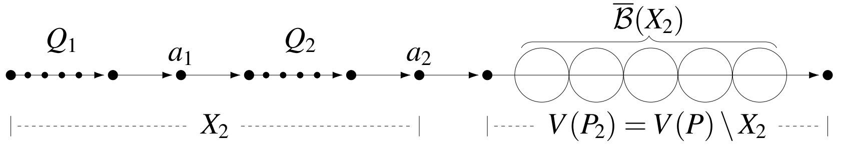


Figure 20 – Illustration of a (2)-split of P. A circle represents an element of the bramble $\overline{\mathcal{B}}(X_2)$.

Now, the proof of Theorem 3.1.16 follows three steps. First, Lemma 3.4.3 states that the set of vertices $\{a_1,\ldots,a_i\}$ of a (i)-split of P is well-linked when $i \leq k$. Thus our goal is to construct a (k)-split of P. Then, Lemma 3.4.4 states that, for $i \geq 0$, we can construct an $(i+1)$-split of P from an (i)-split of P in FPT time if $\mathsf{ord}(\overline{\mathcal{B}}(X_i))$ is large enough. Finally, the proof of Theorem 3.1.16 starts from a (0)-split of P and iterates Lemma 3.4.4 until a (k)-split is constructed.

Lemma 3.4.3. *Let $\mathcal{S}_i$ be an (i)-split of P with $i \in [k]$. Then the set A with $A = \{a_1,\ldots,a_i\}$ is well-linked in D.*

Proof. Let X and Y be disjoint subsets of A such that $|X| = |Y| = r$ for some $r \in [i]$. Suppose, by contradiction, that there is no set of r pairwise internally disjoint paths from X to Y in D. Then, by Menger's Theorem, there is an (X,Y)-separator $S \subseteq V(D)$ such that $|S| \leq r-1$.

Let $Q_{i+1} = P_i$ and $\mathcal{B}_{i+1} = \overline{\mathcal{B}}(X_i)$. By the definition of (i)-splits and our choice of Q_{i+1}, we know that for every $a_j \in A$ with $j \in [i]$, there is a path Q_j ending on the vertex occurring in P before a_j, and a path Q_{j+1} starting on the first vertex occurring in P after a_j (see Figure 20 for an example when $i = 2$). Moreover, we have

$$\mathsf{ord}(\mathcal{B}_{i+1}) \geq g(k) - i\left(\left\lfloor\frac{k}{2}\right\rfloor+1\right)$$

which implies that $\mathsf{ord}(\mathcal{B}_{i+1}) \geq \lfloor k/2\rfloor$ since $i \leq k$.

As $|S| \leq r-1 \leq \lfloor k/2\rfloor - 1$ there must be a path Q_{j+1} such that $a_j \in X\setminus S$ and $S\cap V(Q_{j+1}) = \emptyset$. Furthermore, since S is not large enough to be a hitting set of $\mathcal{B}_{j+1}$, there must be $B \in \mathcal{B}_{j+1}$ such that $S \cap V(B) = \emptyset$. Similarly, there must be an $a_\ell \in Y \setminus S$ such that $S \cap V(Q_\ell) = \emptyset$ and a $B' \in \mathcal{B}_\ell$ such that $S \cap V(B') = \emptyset$.

Now, since $V(Q_{j+1})$ is a hitting set of $\mathcal{B}_{j+1}$, the set S is disjoint from $V(Q_{j+1})$ and from $V(B)$, and $V(B)$ induces a strongly connected subgraph of D, there is in $D \setminus S$ a path from a_j to any vertex in $V(B)$. Similarly, there is a path from any vertex in $V(B')$ to a_ℓ in $D \setminus S$. Finally, since every pair of elements in $\mathcal{B}_T$ intersect, we conclude that there is a path in $D \setminus S$ from a_j to a_ℓ using the path Q_{j+1}, the vertices in $V(B) \cup V(B)'$, and the path Q_ℓ. This contradicts our choice of S, and thus we conclude that every (X,Y)-separator in D must have size at least r, and the result follows by Menger's Theorem. □

Lemma 3.4.4. *Let $\mathcal{S}_i$ be an (i)-split of P with $i \leq k-1$. Then in time $2^{\mathcal{O}(k^2 \log k)} \cdot n^{\mathcal{O}(1)}$ we can construct an $(i+1)$-split of P.*

Proof. For a digraph F, for the sake of notational simplicity, we abbreviate –recall Definition 3.3.3– $\mathcal{B}(V(F))$ and $\overline{\mathcal{B}}(V(F))$ as $\mathcal{B}(F)$ and $\overline{\mathcal{B}}(F)$, respectively, and write $\mathcal{B}(v)$ and $\overline{\mathcal{B}}(v)$ (omitting the braces) for $v \in V(F)$. Let $\mathcal{B}' = \overline{\mathcal{B}}(X_i)$.

The goal is to construct a subpath Q_{i+1} of P starting on the first vertex of P appearing after the vertex a_i (or simply the first vertex of P if $i = 0$) such that

$$\mathsf{ord}(\mathcal{B}'(Q_{i+1})) \geq \left\lfloor \frac{k}{2} \right\rfloor.$$

That is, the order of the bramble containing the elements of $\mathcal{B}$ which are disjoint from X_i while intersecting $V(Q_{i+1})$ is at least $\lfloor k/2 \rfloor$. We start with $V(Q_{i+1}) = \emptyset$. By Inequality 3.1, we have that

$$\mathsf{ord}(\mathcal{B}'(Q_{i+1})) \geq \mathsf{ord}(\mathcal{B}') - \mathsf{ord}(\overline{\mathcal{B}'}(Q_{i+1}))$$

at any point of the procedure. Now, we iteratively grow Q_{i+1}, adding one vertex at a time while testing, at each newly added vertex, whether

$$\mathsf{ord}(\overline{\mathcal{B}'}(Q_{i+1})) \leq g(k) - i\left(\left\lfloor \frac{k}{2} \right\rfloor + 1\right) - 1 - \left\lfloor \frac{k}{2} \right\rfloor.$$

Observe that, when $V(Q_{i+1}) = \emptyset$, we have $\overline{\mathcal{B}'}(Q_{i+1}) = \mathcal{B}'$ and thus

$$\mathsf{ord}(\overline{\mathcal{B}'}(Q_{i+1})) \geq g(k) - i\left(\left\lfloor \frac{k}{2} \right\rfloor + 1\right) > g(k) - i\left(\left\lfloor \frac{k}{2} \right\rfloor + 1\right) - \left\lfloor \frac{k}{2} \right\rfloor.$$

As $\mathcal{B}' = \overline{\mathcal{B}}(X_i)$, we have $\overline{\mathcal{B}'}(Q_{i+1}) = \overline{\mathcal{B}}(X_i \cup V(Q_{i+1}))$ and thus, by Corollary 3.3.6, we can test whether $\mathsf{ord}(\overline{\mathcal{B}'}(Q_{i+1})) \leq s$ in time $2^{\mathcal{O}(k^2 \log k)} \cdot n^{\mathcal{O}(1)}$ for any $s \in [g(k)]$ since $g(k) = \mathcal{O}(k^2)$.

On a negative answer, we add to Q_{i+1} the first vertex of P not contained in $V(Q_{i+1}) \cup X_i$ and repeat the test. On the first time we obtain a positive answer to this test, we set $\mathcal{B}_{i+1} = \mathcal{B}'(Q_{i+1})$, define a_{i+1} to be the first vertex appearing in P after the last vertex of Q_{i+1}, and stop the procedure. In this case, we have that $\mathsf{ord}(\mathcal{B}_{i+1}) \geq \lfloor k/2 \rfloor$ and since $\mathsf{ord}\left(\overline{\mathcal{B}'}(Q_{i+1})\right)$ can decrease by at most one each time we increase by one the size of $V(Q_{i+1})$, this procedure actually ends with $\mathsf{ord}(\overline{\mathcal{B}'}(Q_{i+1})) = g(k) - i(\lfloor k/2 \rfloor + 1) - \lfloor k/2 \rfloor$.

Now, we define $X_{i+1} = X_i \cup V(Q_{i+1}) \cup \{a_{i+1}\}$ and P_{i+1} to be the subpath of P with $V(P_{i+1}) = V(P) \setminus X_{i+1}$. Finally, let $\mathcal{B}^* = \overline{\mathcal{B}'}(Q_{i+1})$. Then by Inequality 3.1,

$$\mathsf{ord}(\mathcal{B}^*(P_{i+1})) \geq \mathsf{ord}(\mathcal{B}^*) - \mathsf{ord}(\overline{\mathcal{B}^*}(P_{i+1}))$$

and observing that $\mathcal{B}^*(P_{i+1}) = \overline{\mathcal{B}}(X_{i+1})$, we conclude that

$$\mathsf{ord}(\overline{\mathcal{B}}(X_{i+1})) \geq g(k) - i\left(\left\lfloor \frac{k}{2} \right\rfloor + 1\right) - \left\lfloor \frac{k}{2} \right\rfloor - 1 = g(k) - (i+1)\left(\left\lfloor \frac{k}{2} \right\rfloor + 1\right),$$

as required, since $\overline{\mathcal{B}^*}(P_{i+1}) = \overline{\mathcal{B}^*}(a_{i+1})$ and thus $\mathsf{ord}(\overline{\mathcal{B}^*}(P_{i+1})) \leq 1$. Then, we output the $(i+1)$-split $\mathcal{S}_{i+1}$ of P_i formed by the sequence of paths $Q_1, \ldots, Q_{i+1}$, the path P_{i+1}, the sequence of brambles $\mathcal{B}_1, \ldots, \mathcal{B}_{i+1}$, the set of vertices $\{a_1, \ldots, a_{i+1}\}$, and the set of vertices X_{i+1}. $\square$

We remark that the bramble $\overline{\mathcal{B}'}(Q_{i+1})$ is used only in the proof of Lemma 3.4.3 and thus we do not need to maintain it during the algorithm. However, if we want to store this information, it suffices to maintain the set T, the set X_i, and the path Q_{i+1} since the bramble $\overline{\mathcal{B}'}(Q_{i+1})$ is equal to the bramble $\mathcal{B}(Q_{i+1})$ in the digraph $D \setminus X_i$. We are now ready to prove Theorem 3.1.15.

Theorem 3.1.15 (Second main contribution). *Let $g(k) = (k+1)(\lfloor k/2 \rfloor + 1) - 1$, D be a digraph and T be a $(g(k)-1, g(k)-1)$-linked set in D with $|T| = 2g(k) - 1$. There is an algorithm running in time $2^{\mathcal{O}(k^2 \log k)} \cdot n^{\mathcal{O}(1)}$ that finds in D a bramble $\mathcal{B}$ of order $g(k)$, a path P that is a hitting set of $\mathcal{B}$, and a well-linked set A of order k such that $A \subseteq V(P)$.*

Proof. Applying Theorem 3.1.9 with input D, we obtain a set $T \subseteq V(D)$ of size $2g(k) - 1$ such that D does not contain a $(T, g(k)-1)$-balanced separator of size $g(k) - 1$. Now, by Lemma 3.3.2, the T-bramble $\mathcal{B}_T$ has order $g(k)$ and, by Lemma 3.4.1, we can find a path P that is a hitting set of $\mathcal{B}_T$ in polynomial time. We start with a trivial (0)-split $\mathcal{S}_0$ of P where $P_0 = P$ and $X_0 = \emptyset$.

For $i \in \{0, \ldots, k-1\}$, we apply Lemma 3.4.4 with input $\mathcal{S}_i$ to obtain an $(i+1)$-split $\mathcal{S}_{i+1}$ of P in time $2^{\mathcal{O}(k^2 \log k)} \cdot n^{\mathcal{O}(1)}$. After the last iteration, we obtained a (k)-split $\mathcal{S}_k$ of P and, by Lemma 3.4.3, the set of vertices $\{a_1,\dots,a_k\}$ of $\mathcal{S}_k$ is well-linked in D and all such vertices are in $V(P)$, as desired. □

By following the remaining of the proof of the Directed Grid Theorem [4], which yields FPT algorithms for all the remaining steps, we can validate Corollary 3.1.16.

3.5 Concluding remarks

The main consequence of our results is an FPT algorithm with parameter k that either produces an arboreal decomposition of width at most $f(k)$ for a digraph D or constructs a cylindrical grid of order k as a butterfly minor of D, for some computable function $f(k)$. This is achieved by adapting some of the steps used in the proof of the Directed Grid Theorem from Kawarabayashi and Kreutzer [4].

For the first possible output of this algorithm, we improve on a result from [24] by providing an FPT algorithm with parameter k that either produces an arboreal decomposition of a digraph D with width at most $3k-2$, or concludes that D has a haven of order k. As a tool to prove this result, we show how to solve the BALANCED SEPARATOR problem, which generalizes the problem of finding balanced separators, in FPT time with parameter $|T|$. Since in the undirected case balanced separators are strongly related to the tree-width of undirected graphs, and the only result for balanced separators in the directed case considered only a relaxed version of the problem (see [58, Chapter 6]), we consider this result to be of its own interest. We remark that, in a loopless digraph, a $(V(D),0)$-balanced separator is exactly a directed feedback vertex set and thus BALANCED SEPARATOR generalizes DIRECTED FEEDBACK VERTEX SET (DFVS). We leave open the question of whether BALANCED SEPARATOR is FPT when parameterized by the size of the solution.

Although it is possible to construct a bramble $\mathcal{B}$ of order $\lfloor k/2 \rfloor$ from a haven of order k, this construction is not *efficient* in general, in the sense that we must go through all elements of $\mathcal{B}$ to verify whether a given set X is a hitting set of $\mathcal{B}$. Motivated by this, we consider a definition of brambles, which we call T-brambles, which naturally occur in digraphs of large directed tree-width that are better to work with in a number of ways. For instance, by reducing to the problem of computing balanced separators for T, we show how to compute hitting sets of T-brambles in FPT time when parameterized by $|T|$.

We use our results for T-brambles in digraphs of large tree-width to show how to find, in FPT time with parameter k, a path that is a hitting set of a T-bramble $\mathcal{B}_T$ of order $(k+1)(\lfloor k/2 \rfloor + 1)$ and a well-linked set of size k that is contained in this path. This is the second step that we change in the proof of the Directed Grid Theorem [4]. From this point forward, the remaining steps in the proof yield FPT algorithms.

Kreutzer and Ordyniak [63] and Ganian et al. [64] showed that many important problems in digraphs remain hard when restricted to digraphs of bounded directed tree-width. In particular, Kreutzer and Ordyniak [63] showed that the DIRECTED FEEDBACK VERTEX SET problem is NP-complete even when restricted to digraphs of directed tree-width at most five. However, some open problems in digraphs may benefit from an approach resembling Bidimensionality using our FPT algorithm for the Directed Grid Theorem. For example, Bezáková et al. [65] asked whether the LONGEST DETOUR problem in digraphs could be solved by using the Directed Grid Theorem. To provide more potential applicability of our results, we briefly discuss the parameterized tractability of the DFVS problem.

Chen et al. [60] provided an algorithm running in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ for the DFVS problem, where k is the size of the solution. It is an open problem whether the dependency on this parameter can be improved to $2^{\mathcal{O}(k)}$, and whether DFVS admits a polynomial kernel, even in planar digraphs. Bonamy et al. [66] showed that, when parameterized by the tree-width t of the underlying graph, DFVS is solvable in time $2^{\mathcal{O}(t \log t)} \cdot n^{\mathcal{O}(1)}$ in general digraphs and the dependency on the parameter is improved to $2^{\mathcal{O}(t)}$ when restricted to planar digraphs. When parameterized by the feedback vertex set number of the underlying graph, Bergougnoux et al. [67] showed that DFVS admits a polynomial kernel in general digraphs, and a linear kernel in digraphs that are embeddable on surfaces of bounded genus.

On the one hand, DFVS remains hard even when restricted to digraphs of directed tree-width at most five [63], but on the other hand both of the aforementioned parameters related to the underlying graph are individually stronger than the directed tree-width of the input digraph and, by the Directed Grid Theorem [4], every positive instance of DFVS parameterized by the size k of the solution occurs in a digraph of bounded directed tree-width: since a cylindrical grid of order r contains a set of r vertex-disjoint cycles and butterfly contractions do not generate new paths, the minimum size of a feedback vertex set of a digraph D is at least the order of the largest cylindrical grid that is as a butterfly minor of D. Now, by Corollary 3.1.16, in FPT time with parameter k we can either find a certificate that the considered instance of DFVS is negative (a cylindrical grid of order $k+1$ that is a butterfly minor of the input digraph), or produce an arboreal decomposition of the input digraph of width at most $f(k)$, for some computable function $f : \mathbb{N} \to \mathbb{N}$.

Thus, it is sensible to ask whether similar or improved results for DFVS (when parameterized by the tree-width or the feedback vertex set number of the underlying graph, as previously mentioned) can be proved if we consider that the input digraph has bounded directed tree-width, since by the above discussion we can restrict instances of DFVS to this class of digraphs.

One could also consider the tractability of hard problems in digraphs of bounded directed tree-width under stronger parameterizations. For example, in Chapter 4 we show that a relaxation for the DIRECTED DISJOINT PATHS problem, a notoriously hard problem in digraphs, admits a kernelization algorithm for some choices of parameters. In this spirit, it seems plausible that combining directed tree-width with other parameters may lead to FPT algorithms for hard problems, and in this context the FPT algorithm presented in this chapter may become handy.

It is worth mentioning that Giannopoulou et al. [68] recently provided an analogous version of the Flat Wall Theorem [1] for directed graphs, which may have interesting algorithmic applications when combined with our results.

Finally, the attempts to obtain a Bidimensionality theory for directed graphs, such as the one presented by Dorn et al. [69], are so far less satisfying that the undirected version, from the point of view of generality and efficiency of the obtained algorithms. We hope that our FPT version of the Directed Grid Theorem will have a relevant role in an eventual Bidimensionality theory for directed graphs.

4 THE DISJOINT ENOUGH PATHS PROBLEM

Robertson and Seymour [1] showed, in their seminal work on graph minors, that DISJOINT PATHS is FPT when parameterized by the number k of requests. The DIRECTED DISJOINT PATHS (DDP) problem, however, turns out to be significantly harder: Fortune et al. [2] showed that the problem is NP-complete even for fixed $k = 2$. In order to obtain positive results, a common approach has been to consider restricted input digraphs. For instance, it is also shown in [2] that DDP is XP in DAGs with parameter k. For some time the question of whether this could be improved to an FPT algorithm remained open, but a negative answer was given by Slivkins [3]: DDP is W[1]-hard in DAGs with this parameter k. Johnson et al. [24] extended the previous result for DAGs by showing an $n^{\mathcal{O}(k+w)}$ algorithm for DDP in digraphs with directed tree-width (see Section 2.3) at most w. Another restriction considered in the literature is to ask for the underlying graph of the input digraph to be planar. Under this restriction, Schrijver [28] provided an XP algorithm for DDP with parameter k, which was improved a long time afterwards to an FPT algorithm by Cygan et al. [29].

A natural relaxation for the DIRECTED DISJOINT PATHS problem is to allow for vertex and/or edge congestion. Namely, in the DIRECTED DISJOINT PATHS WITH CONGESTION problem (DDPC for short, or DDPC-c if we want to specify the value of the congestion), the task is to find a collection of paths satisfying the k requests such that no vertex in the graph occurs in more than c paths of the collection. Amiri et al. [27] considered the tractability of this problem when restricted to DAGs. By a simple local reduction to the general version and the hardness result by Slivkins [3], they showed that DDPC-c in DAGs is W[1]-hard for every fixed $c \geq 1$. The authors also proved that DDPC-c admits an XP algorithm with parameter d in DAGs, where $d = k - c$. Together with the result by Johnson et al. [24], this simple reduction presented in [27] is also sufficient to show that DDPC-c also admits an XP algorithm with parameters k and w for every fixed $1 \leq c \leq n-1$ in digraphs with directed tree-width at most w, and the same result also holds when we allow for congestion on the edges.

Motivated by Thomassen's proof [70] that DDP remains NP-complete for $k = 2$ when restricted to β-strongly connected digraphs, for any integer $\beta \geq 1$, Edwards et al. [71] recently considered the DDPC-2 problem (this version of the problem is usually called *half-integral* in the literature) and proved, among other results, that it can be solved in time $n^{f(k)}$ when restricted to $(36k^3 + 2k)$-strongly connected digraphs.

Kawarabayashi et al. [72] considered the following asymmetric version of the DDPC-

4 problem: the task is to either find a set of paths satisfying the requests with congestion at most four, or to conclude that no set of pairwise vertex-disjoint paths satisfying the requests exists. In other words, we ask for a solution for DDPC-4 or a certificate that there is no solution for DDP. They proved that this problem admits an XP algorithm with parameter k in general digraphs, and claimed –without a proof– that Slivkins' reduction [3] can be modified to show that it is W[1]-hard in DAGs. In their celebrated proof of the Directed Grid Theorem, Kawarabayashi and Kreutzer [4] claimed that an XP algorithm can be also obtained for the asymmetric version with congestion at most three. To the best of our knowledge, the existence of an XP algorithm in general digraphs for the DDPC-2 problem, or even for its asymmetric version, remains open.

Summarizing, the existing positive results in the literature for parameterizations and/or relaxations of the DIRECTED DISJOINT PATHS problem in *general digraphs* are quite scarce.

Our approach, results, and techniques. In this chapter, we propose another congestion metric for DDP. In contrast to the usual relaxations discussed above, which focus on a *local* congestion metric that applies to every vertex, our approach considers, on top of local congestion, a *global* congestion metric: we want to keep control of how many vertices (a global metric) appear in "too many" paths (a local metric) of the solution. That is, we want the paths to be such that "most" vertices of the graph do not occur in too many paths, while allowing for any congestion in the remaining vertices. In the particular case where we do not allow for local congestion, we want the paths to be pairwise vertex-disjoint not in the whole graph, but in an unspecified part of size prescribed by a parameter; this is why we call such paths "disjoint enough".

Formally, in the DISJOINT ENOUGH DIRECTED PATHS (DEDP) problem, we are given a set of requests $\{(s_1,t_1),\ldots,(s_k,t_k)\}$ in a digraph D and two non-negative integers c and s, and the task is to find a collection of paths $\{P_1,\ldots,P_k\}$ such that each P_i is a path from s_i to t_i in D and at most c vertices of D occur in more than s paths of the collection. If $s = 1$, for instance, we ask for the paths to be pairwise vertex-disjoint in at least $n - c$ vertices of the graph, and allow for at most c vertices occurring in two or more paths. Choosing $c = 0$ and $s = 1$, DEDP is exactly the DDP problem and, choosing $s = 0$, DEDP is exactly the STEINER NETWORK problem (see [14] for its definition).

By a simple reduction from the DIRECTED DISJOINT PATHS WITH CONGESTION problem, it is easy to prove that DEDP is NP-complete for fixed $k \geq 3$ and $s \geq 1$, even if c is large with respect to n, namely at most $n - n^{\alpha}$ for some real value $0 < \alpha \leq 1$, and W[1]-hard in

DAGs with parameter k. By applying the framework of Johnson et al. [24], we give an $n^{\mathcal{O}(k+w)}$ algorithm to solve DEDP in digraphs with directed tree-width at most w.

The fact that DEDP is NP-complete for fixed values of $k = 2$, $c = 0$, and $s = 1$ [2] motivates us to consider the "dual" parameter $d = n - c$. That is, instead of bounding from above the number of vertices of D that lie in the intersection of many paths of a collection satisfying the given requests, we want to bound from below the number of vertices that occur only in few paths of the collection. Formally, we want to find $X \subseteq V(D)$ with $|X| \geq d$ such that there is a collection of paths $\mathcal{P}$ satisfying the given requests such that every vertex in X is in at most s paths of the collection. We first prove, from a reduction from the INDEPENDENT SET problem, that DEDP is W[1]-hard with parameter d for every fixed $s \geq 0$, even if the input graph is a DAG and all source vertices of the request set are the same.

Our main contribution consists of positive algorithmic results for this dual parameterization. On the one hand, we give an algorithm for DEDP running in time $\mathcal{O}(n^d \cdot k^{d \cdot s})$. This algorithm is not complicated, and basically performs a brute-force search over all vertex sets of size d, followed by k connectivity tests in a digraph D' obtained from D by an appropriate local modification. On the other hand, our most technically involved result is a kernel for DEDP with at most $d \cdot 2^{k-s} \cdot \binom{k}{s}$ non-terminal vertices. This algorithm first starts by a reduction rule that eliminates what we call *congested* vertices; we say that the resulting instance is *clean*. We then show that if D is clean and sufficiently large, and $k = s + 1$, then the instance is positive and a solution can be found in polynomial time. This fact is used as the base case of an iterative algorithm. Namely, we start with the original instance and proceed through $k - s + 1$ iterations. At each iteration, we choose one path from some s_i to its destination t_i such that a large part of the graph remains unused by any of the pairs chosen so far (we prove that such a request always exists) and consider only the remaining requests for the next iteration. We repeat this procedure until we arrive at an instance where the number of requests is exactly $s + 1$, and use the base case to output a solution for it. From this solution, we extract in polynomial time a solution for the original instance, yielding a kernel of the claimed size.

Since positive results for the DIRECTED DISJOINT PATHS problem are not common in the literature, especially in general digraphs, we consider our algorithmic results to be of particular interest. Furthermore, the kernelization algorithm also brings good news for the STEINER NETWORK problem: when $s = 0$ Feldmann and Marx in [14] showed that the tractability of the STEINER NETWORK problem when parameterized by the number of requests depends on how the requests are structured. Our result adds to the latter by showing that the problem remains FPT if we drop this structural condition on the request set but add d, the number of vertices occurring in at most s paths of the solution, as a parameter. More details can be found in Section 4.1.

For reference, we include here another copy of Table 1, showing a summary of our algorithmic and complexity results, which altogether provide an accurate picture of the parameterized complexity of the DEDP problem for distinct choices of the parameters.

k	d	s	w	Complexity
fixed ≥ 3	$\Omega(n^{\alpha})$	fixed ≥ 1	—	NP-complete (Theorem 4.2.1)
parameter	$\Omega(n^{\alpha})$	fixed ≥ 1	0	W[1]-hard (Theorem 4.2.1)
input	parameter	fixed ≥ 0	—	W[1]-hard (Theorem 4.2.2)
parameter	—	—	parameter	XP (Theorem 4.3.10)
input	parameter	parameter	—	XP (Theorem 4.3.12)
parameter	parameter	parameter	—	FPT (Theorem 4.3.21)

Table 1 – Summary of hardness and algorithmic results for distinct choices of the parameters. A horizontal line in a cell means no restrictions for that case. In all cases, we have that $c = n - d$ and $0 < \alpha \leq 1$.

This chapter is organized as follows. In Section 4.1 we present some preliminaries and formally define the Disjoint Enough Directed Paths problem. We provide the hardness results in Section 4.2 and the algorithms in Section 4.3. We conclude the chapter in Section 4.4 with some open questions for further research.

4.1 Preliminaries

Before defining the problem, we remind the reader of Definition 2.4.2.

Definition 2.4.2 (Requests and satisfying collections). *Let D be a digraph and $\mathcal{P}$ be a collection of paths of D. A* request *in D is an ordered pair of vertices of D. For a request set $I = \{(s_1,t_1),(s_2,t_2),\ldots,(s_k,t_k)\}$, we say that the vertices $\{s_1,s_2,\ldots,s_k\}$ are* source *vertices and that $\{t_1,t_2,\ldots,t_k\}$ are* target *vertices, and we refer to them as $S(I)$ and $T(I)$, respectively. We say that $\mathcal{P}$* satisfies *I if $\mathcal{P} = \{P_1,\ldots,P_k\}$ and P_i is a path from s_i to t_i, for $i \in [k]$.*

Again we remark that a request set may contain many copies of the same pair, and that when considering the union of two or more requests, we keep all such copies in the resulting request set. For instance, if $I_1 = \{(u_1,v_1)\}$ and $I_2 = \{(u_1,v_1),(u_2,v_2)\}$ then $I_1 \cup I_2 =$

$\{(u_1,v_1),(u_1,v_1),(u_2,v_2)\}$, and this indicates that a collection of paths satisfying this request set must contain two paths from u_1 to v_1. The DEDP problem is defined as follows.

DISJOINT ENOUGH DIRECTED PATHS (DEDP)

Input: A digraph D, a request set I of size k, and two non-negative integers c and s.

Output: A collection of paths $\mathcal{P}$ satisfying I such that at most c vertices of D occur in at least $s+1$ paths of $\mathcal{P}$ and all other vertices of D occur in at most s paths of $\mathcal{P}$.

Unless stated otherwise, we consider $d = n - c$ for the remaining of this chapter. Intuitively, c imposes an upper bound on the size of the "**c**ongested" part of the solution, while d imposes a lower bound on the size of the "**d**isjoint" part. For a parameterized version of DEDP, we sometimes include the parameters before the name. For instance, we denote by (k,d)-DEDP the DISJOINT ENOUGH DIRECTED PATHS problem with parameters k and d.

Notice that if $c \geq n$ or $s \geq k$, the problem is trivial since every vertex of the graph is allowed to be in all paths of a collection satisfying the requests, and thus we need only to check for connectivity between the given pairs of vertices. Furthermore, if there is a pair (s_i,t_i) in the request set such that there is no path from s_i to t_i in the input digraph D, the instance is negative. Thus we henceforth assume that $c < n$, that $s < k$, and that there is a path from s_i to t_i in D for every pair (s_i,t_i) in the set of requests.

Choosing the values of k, d, and s appropriately, we show in Table 2 that the DEDP problem generalizes several problems in the literature.

Parameters	Equivalent to	Complexity
$d = n$, $s = 1$	DISJOINT PATHS	NP-complete for $k = 2$ [2]
$d = n$, $s \geq 2$	DISJOINT PATHS WITH CONGESTION	W[1]-hard with parameter k [3]
$d \geq 1$, $s = 0$	STEINER NETWORK	FPT with parameters k and d

Table 2 – Summary of related problems and complexity results.

The last line of Table 2 is of particular interest, and we focus on it in the next two paragraphs. In the STEINER NETWORK problem, we are given a digraph D and a request set I and we are asked to find an induced subgraph D' of D with minimum number of vertices such that D' admits a collection of paths satisfying I. For a request set I in a digraph D, let $D(I)$ be the digraph with vertex set $S(I) \cup T(I)$ and edge set $\{(s,t) \mid (s,t) \in I\}$. The complexity landscape of the STEINER NETWORK problem when parameterized by the size of the request set was given by Feldmann and Marx [14]. They showed that the tractability of the problem depends on $D(I)$.

Namely, they proved that if $D(I)$ is close to being a *caterpillar*, then the STEINER NETWORK problem is FPT when parameterized by $|I|$, and W[1]-hard otherwise. When parameterized by the size of the solution, Jones et al. [73] showed that the STEINER NETWORK problem is FPT when $D[I]$ is a star whose edges are all oriented from the unique source and the underlying graph of the input digraph excludes a topological minor, and W[2]-hard on graphs of degeneracy two [73].

Our algorithmic results for DEDP for the particular case $s = 0$ yield an FPT algorithm for another parameterized variant of the STEINER NETWORK problem. In this case, we want to decide whether D admits a large set of vertices whose removal does not disconnect any pair of requests. That is, we want to find a set $X \subseteq V(D)$ with $|X| \geq d$ such that $D \setminus X$ contains a collection of paths satisfying I. In Theorem 4.3.21 we give an FPT algorithm (in fact, a kernel) for this problem with parameters $|I|$ and d. We remark that this tractability does not depend on $D(I)$.

4.2 Hardness results for DEDP

In this section we provide hardness results for the DEDP problem. Namely, we first provide in Theorem 4.2.1 a simple reduction from DISJOINT PATHS WITH CONGESTION, implying NP-completeness for fixed values of k, c, d and W[1]-hardness in DAGs with parameter k. We then prove in Theorem 4.2.2 that DEDP is W[1]-hard in DAGs with parameter d.

As mentioned in [73], the STEINER NETWORK problem is W[2]-hard when parameterized by the size of the solution (as a consequence of the results of [74]). Hence (c)-DEDP is W[2]-hard for fixed $s = 0$. As discussed in the introduction, the DIRECTED DISJOINT PATHS problem is NP-complete for fixed $k = 2$ [2] and W[1]-hard with parameter k in DAGs [3]. Allowing for vertex congestion does not improve the tractability of the problem: DISJOINT PATHS WITH CONGESTION parameterized by the number of requests is also W[1]-hard in DAGs for every fixed congestion $c \geq 1$, as observed in [27]. When $c = 0$ and $s \geq 1$, DEDP is equivalent to the DIRECTED DISJOINT PATHS WITH CONGESTION problem and thus the aforementioned bounds also apply to it. In the following theorem we complete this picture by showing that DEDP is NP-complete for fixed $k \geq 3$ and $s \geq 1$, even if c is quite large with respect to n (note that if $c = n$ all instances are trivially polynomial), namely for c as large as $n - n^{\alpha}$ with α being any fixed real number such that $0 < \alpha \leq 1$. The same reduction also allows to prove W[1]-hardness in DAGs with parameter k. The idea is, given the instance of DDPC with input digraph D, build an instance of DEDP where the "disjoint" part corresponds to the original instance, and the "congested" consists of c new vertices that are necessarily used by $s+1$ paths of any solution. In this process, we generate an instance of DEDP in a digraph D' with $|V(D')| = n = d+c$ and $d = |V(D)|$. This is why we restrict the value of d to be of the form n^{α}, but not smaller: if we ask d to be "too small", $d = \log n$ for example, our procedure would generate an instance of DEDP such that the size of the "disjoint part" d satisfies $d = \log(d+c)$ which in turn implies that the size of this instance would be exponential on the size of the original instance of DDPC.

Theorem 4.2.1. *Let $0 < \alpha \leq 1$ and $d : \mathbb{N} \to \mathbb{N}$ with $d(n) = \Omega(n^{\alpha})$. Then, for $c = n - d(n)$,*

(i) DEDP *is* NP*-complete for every fixed $k \geq 3$ and $s \geq 1$; and*

(ii) (k)-DEDP *is* W[1]*-hard in DAGs for every fixed $s \geq 1$.*

Proof. We prove items **(i)** and **(ii)** at the same time by a simple reduction from the Directed Disjoint Paths with Congestion (DDPC) problem. Given an instance (D, I, k, s) of DDPC, we output an equivalent instance $(D', I', k+s, c, s)$ of DEDP that does not generate any new cycles and such that the size $d(|V(D')|)$ of the disjoint part of the new instance is equal to $|V(D)|$, with $d(n)$ as in the statement of the theorem. Since DDP, which is exactly the DDPC problem with congestion one, is NP-complete for fixed $k \geq 2$ [2] and k-DDPC is W[1]-hard in DAGs [27], our reduction implies that DEDP with $c = n - d(n)$ is NP-complete for fixed $k \geq 3$ and $s \geq 1$, and W[1]-hard in DAGs with parameter k and any fixed $s \geq 1$. We can assume that $c \geq 1$ since DEDP is exactly DDPC when $c = 0$ and $s \geq 1$ (as discussed previously).

Formally, let (D, I, k, s) be an instance of DDPC with $I = \{(s_1,t_1),(s_2,t_2),\ldots,(s_k,t_k)\}$ and choose $i \in [k]$ arbitrarily. We construct an instance of DEDP as follows. Let D' be a digraph on vertex set $V(D)$ together with new vertices $\{v_1,\ldots,v_c\}$. Add to $E(D')$ all edges from $E(D)$ plus the set (v_j, v_{j+1}) for $j \in [c-1]$. Finally, add to I' all pairs in $(I \setminus \{s_i,t_i\})$, the pair (s_i, v_c), and s copies of the pair (v_1, v_c). Figure 21 illustrates this construction. It is easy to verify that

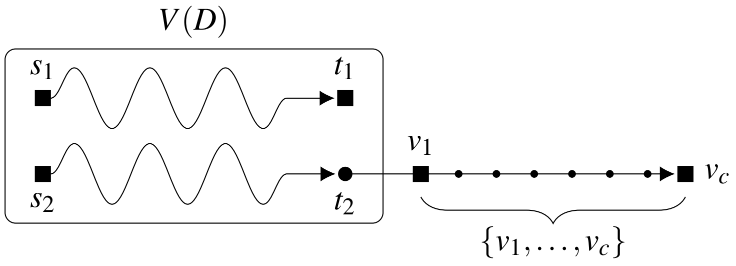


Figure 21 – Example of the construction from Theorem 4.2.1 with $k=2$, $s=1$, and $i=2$. Source and target vertices are represented by square vertices in the figure.

(D,I,k,s) is positive if and only if the instance $(D',I',k+s,c,s)$ of DEDP is positive; we provide the formal proof for completeness.

For the necessity, let $\mathcal{P}$ be a solution for (D,I,k,s), where $\mathcal{P} = \{P_1,P_2,\ldots,P_k\}$. Let P_i' be the path in D' formed by a copy of P_i together with the path from t_i to v_c and $\mathcal{Q}$ be the collection formed by s copies of the unique path in D' from v_1 to v_c. Now, $(\mathcal{P}\setminus\{P_i\})\cup\{P_i'\}\cup\{\mathcal{Q}\}$ is a solution for $(D',I',k+s,c,s)$ as every vertex in $V(D)$ occurs in at most s paths of $\mathcal{P}$. Similarly, a solution $\mathcal{P}'$ for $(D',I',k+s,c,s)$ yields a solution for (D,I,k,s): as all vertices in $\{v_1,\ldots,v_c\}$ occur in $s+1$ paths of $\mathcal{P}'$, every vertex in $V(D)$ can appear in at most s paths of the collection. We now show that the construction runs in polynomial time for our choice of $d(n)$.

By definition, if $d(n) = \Omega(n^\alpha)$ then there are positive constants a and n_0 such that $d(n) \geq a \cdot n^\alpha$ for all $n \geq n_0$, and we can assume that $|V(D)| \geq n_0$ since DDPC is solvable in constant time on instances of fixed size through a brute force algorithm. In the construction, we generate an instance of DEDP in a digraph D' such that $|V(D')| = |V(D)| + c$ and thus, to satisfy $d(|V(D')|) = |V(D)|$, we want to choose c such that $d(|V(D)|+c) = |V(D)|$. Now, by the choice of $d(n)$ we get that $d(|(V(D')|+c) \geq a\cdot(|V(D)|+c)^\alpha$ and therefore our goal is satisfied choosing c such that $|V(D)| \geq a\cdot(|V(D)|+c)^\alpha$ or, equivalently, $c \leq (|V(D)|/a)^{1/\alpha} - |V(D)| = \mathcal{O}(|V(D)|^{1/\alpha})$. □

Next, we show that (d)-DEDP is W[1]-hard, even when the input graph is acyclic and all source vertices of the request set are the same. The reduction is from the INDEPENDENT SET problem parameterized by the size of the solution, which is W[1]-hard [6, 34].

Theorem 4.2.2. *The* DEDP *problem is* W[1]*-hard with parameter d for every fixed $s \geq 0$, even when the input graph is acyclic and all source vertices in the request set are the same.*

Proof. Let (G,d) be an instance of the INDEPENDENT SET problem, in which we want to decide whether the (undirected) graph G contains an independent set of size at least d, and s be a non-negative integer. Let V^E be the set $\{v_e \mid e \in E(G)\}$ and D a directed graph with vertex set $V(G)\cup\{r\}\cup V^E$. Add to D the following edges:

- for every $v \in V(G)$, add the edge (r,v); and
- for every edge $e \in E(G)$ with endpoints u and w, add the edges (u,v_e) and (w,v_e).

Finally, for every $v_e \in V^E$, add $2s+1$ copies of the pair (r,v_e) to I. Figure 22 illustrates this construction.

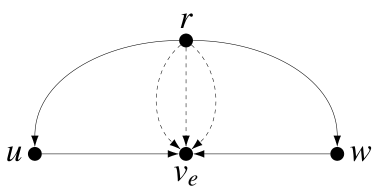


Figure 22 – Example of the construction from Theorem 4.2.2 with $s = 1$ and $e = (u,w)$. A dashed line indicates a request in I.

Notice that each vertex v_e of D associated with an edge E of D has out-degree zero in D and r has in-degree zero. Moreover, every edge of D has as extremity either r or a vertex of the form v_e. Thus D is a acyclic, as desired. Furthermore $|S(I)| = 1$ since all of its elements are of the form (r, v_e), for $e \in E(G)$. We now show that (G,d) is positive if and only if (D,I,k,c,s) is positive, where $k = |I| = m \cdot (2s+1)$.

For the necessity, let X be an independent set of size d in G. Start with a collection $\mathcal{P} = \emptyset$. We classify the edges of G into two sets: the set E_1 containing all edges with both endpoints in $V(G) \setminus X$, and the set E_2 containing all edges with exactly one endpoint in X. Now, for each $e \in E_1$, chose arbitrarily one endpoint u of e and add to $\mathcal{P}$ $2s+1$ copies of the path in D from r to v_e using u. For each $e \in E_2$ with $e = (u,w)$ and $w \notin X$, add to $\mathcal{P}$ $2s+1$ copies of the path in D from r to v_e using w. Since X is an independent set, no vertex in X occurs in any path of $\mathcal{P}$, and since $E(G) = E_1 \cup E_2$, $\mathcal{P}$ satisfies I and the necessity follows as $c = n - d$.

Let $\mathcal{P}$ be a solution for (D,I,k,c,s) and $X \subseteq V(D)$ be a set of vertices with $|X| = d$ and such that each vertex of X occurs in at most s paths of $\mathcal{P}$. Such choice is possible since $d = n - c$. For contradiction, assume that X is not an independent set in G. Then there is an edge $e \in E(G)$ with $e = (u,w)$ and $u,w \in X$, and $2s+1$ copies of the request (r, v_e) in I. Thus each path satisfying one of those requests uses u or w, but not both, and therefore either u or w occurs in at least $s+1$ paths of $\mathcal{P}$, a contradiction. We conclude that X is an independent set in G and the sufficiency follows. $\square$

4.3 Algorithms for DEDP

In this section we focus on algorithmic results for DEDP. In Theorem 4.2.1 we showed that DEDP is NP-complete for every fixed $k \geq 3$ and a large range of values of c. We also showed that considering only d as a parameter is still not enough to improve the tractability of the problem: Theorem 4.2.2 shows that (d)-DEDP is W[1]-hard in DAGs even if all requests share the same source. Thus DEDP is as hard as the DIRECTED DISJOINT PATHS problem when k is *not* a parameter. Here we show that, similarly to the latter, DEDP admits an XP algorithm when parameterized by the number of requests and the directed tree-width of the input digraph. Then, we show how the tractability of DEDP improves when we consider stronger parameterizations including d. Namely, we show that DEDP is XP with parameters d and s (cf. Theorem 4.3.12), and FPT with parameters k and d (hence s as well, since we may assume that $k > s$ as discussed in Section 4.1; cf. Theorem 4.3.21). It is worth mentioning that this kind of *dual parameterization* (remember that $d = n - c$) has proved useful in order to improve the tractability of several notoriously hard problems (cf. for instance [27, 75–78]).

In Section 4.3.1 we apply the ideas and results used by Johnson et al [24] to show an XP algorithm for the DIRECTED DISJOINT PATHS problem parameterized by the number of requests in digraphs of bounded directed tree-width to show that a similar result holds for DEDP. In Section 4.3.2 we show our algorithms for parameterizations of DEDP including d as a parameter.

4.3.1 *An* XP *algorithm with parameters* k *and* $\mathsf{dtw}(D)$

The algorithm consists of dynamic programming along an arboreal decomposition of the input graph. Following the notation used by Johnson et al. [24], we refer to the information we want to compute at every step of the algorithm as an *itinerary*. We provide a formal definition for an itinerary for DEDP later. We recall that a set of vertices S is w-guarded if S is Z-guarded for some Z with $|Z| \leq w$ (cf. Definition 2.3.1).

Johnson et al. [24] provided two conditions that, if satisfied by a given problem, are sufficient to provide an XP algorithm for it in digraphs with bounded directed tree-width. More precisely, for a digraph D with $\mathsf{dtw}(D) = w$, they ask that there is a real number α depending on w (and possibly some parameters of the problem, if any) and two algorithms satisfying the following conditions.

Condition 4.3.1 (Johnson et al. [24]). *Let A, B be two disjoint subsets of $V(D)$ such that there are no edges in D with head in A and tail in B. Then an itinerary for $A \cup B$ can be computed from an itinerary for A and an itinerary for B in time $\mathcal{O}(n^{\alpha})$.*

Condition 4.3.2 (Johnson et al. [24]). *Let A, B be two disjoint subsets of $V(D)$ such that A is w-guarded and $|B| \leq w$. Then an itinerary for $A \cup B$ can be computed from an itinerary for A and an itinerary for B in time $\mathcal{O}(n^{\alpha})$.*

Using this notation, the following theorem says how to compute an itinerary for $V(D)$.

Theorem 4.3.3 (Johnson et al. [24]). *Provided that Conditions 4.3.1 and 4.3.2 hold, there is an algorithm running in time $\mathcal{O}(n^{\alpha+1})$ that receives as input a digraph D and an arboreal decomposition for D with width at most w and outputs an itinerary for $V(D)$.*

In [24] an XP algorithm for the DIRECTED DISJOINT PATHS problem in digraphs of bounded directed tree-width is given as an example of application of the aforementioned tools, and a similar approach is claimed to work for the HAMILTON PATH, HAMILTON PATH WITH PRESCRIBED ENDS, EVEN CYCLE THROUGH A SPECIFIED VERTEX problems, and others. We follow their ideas to provide an XP algorithm for (k,w)-DEDP, where w is the directed tree-width of the input digraph. The main idea, formalized by the following definition and lemma, is that the number of weak components in the digraph formed by the union of the paths in a collection $\mathcal{P}$ satisfying the request set is bounded by a function depending on k and w only. Thus we can guess how the paths in $\mathcal{P}$ cross a set of vertices A that is w-guarded and use an arboreal decomposition of the input digraph to propagate this information in a dynamic programming scheme. We use the following definition.

Definition 4.3.4. *Let D be a digraph and $\mathcal{P}$ be a collection of paths in D. We denote by $D(\mathcal{P})$ the digraph formed by the union of all paths in $\mathcal{P}$.*

Definition 4.3.5 (Limited collections). *Let I be a request set in a digraph D with $|I| = k$ and $\mathcal{P}$ be a collection of paths satisfying I.We say that $\mathcal{P}$ is (k,w,S)-*limited*, for some $S \subseteq V(D)$, if $D(\mathcal{P}) \subseteq D[S]$ and for every w-guarded set $S' \subseteq S$, the digraph induced by $V(D(\mathcal{P})) \cap S'$ has at most $(w+1)\cdot k$ weak components.*

The following lemma is inspired by [24, Lemma 4.5] and is key to the algorithm.

Lemma 4.3.6. *Let I be a request set of size k in a digraph D and w be an integer. Then every collection of paths $\mathcal{P}$ satisfying I is (k,w,S)-limited for every $S \subseteq V(D)$ containing all paths in $\mathcal{P}$.*

Proof. Let $k = |I|$ and S be as in the statement of the lemma and S' be a w-guarded subset of S. By the definition of w-guarded sets, there is a set $Z \subseteq V(D)$ with $|Z| \leq w$ such that S' is Z-guarded. For $i \in [k]$, let $\mathcal{Q}_i$ be the collection of paths formed by the subpaths of P_i intersecting S'. Thus,

$D(\mathcal{Q}_i)$ consists of the union of subpaths of P_i. Let q_i be the number of weak components of $D(\mathcal{Q}_i)$. Since S' is Z-guarded, each subpath of P_i linking two distinct weak components of $D(\mathcal{Q}_i)$ must intersect Z. Thus, $|V(P_i) \cap Z| \geq q_i - 1$ and $q_i \leq w+1$ since a vertex of Z can be in all paths of $\mathcal{P}$. We conclude that $\sum_{i \in [k]} q_i \leq (w+1) \cdot k$, as desired. □

We now formally define an itinerary for DEDP. From this point forward, we say that a request set I in a digraph D is *contained* in A if every vertex occurring in I is contained in A.

Definition 4.3.7 (Itinerary). *Let Γ be an instance of* DEDP *with $\Gamma = (D, I, k, c, s)$, $A \subseteq V(D)$, and $\mathcal{I}_A$ be the set of all request sets on D which are contained in A. For an integer w, a* (Γ, w)-itinerary for *A is a function $f_A : \mathcal{I}_A \times \mathbb{N} \to \{0,1\}$ such that $f_A(I', c') = 1$ if and only if*

(i) *$k' \leq (w+1) \cdot k$, for $k' = |I'|$;*

(ii) *$c' \leq c$; and*

(iii) *the instance $(D[A], I', k', c', s)$ of* DEDP *is positive.*

With this notation, an instance (D, I, k, c, s) is positive if and only if $f_{V(D)}(I, c') = 1$ for some $c' \leq c$. We now provide algorithms satisfying Conditions 4.3.1 and 4.3.2 for the given definition of an itinerary for DEDP. By Lemma 4.3.6, we need to consider only request sets of size at most $(w+1) \cdot k$ whenever the input digraph has directed tree-width at most w in the following lemmas. We follow the proofs given by Johnson et al. [24], adapting them to our case. For every $t \in [n]$, the authors show how to compute a solution containing at most t vertices for a given instance of the DIRECTED DISJOINT PATHS problem, if one exists, or to decide that no such solution exists. We drop this demand in our algorithm, and instead include the restriction on the congestion c.

Lemma 4.3.8. *Let Γ be an instance of* DEDP *with $\Gamma = (D, I, k, c, s)$ and A, B be disjoint subsets of $V(D)$ such that there are no edges in D with head in A and tail in B. Then a (Γ, w)-itinerary for $A \cup B$ can be computed from itineraries for A and B in time $\mathcal{O}(n^{4(w+1) \cdot k + 3})$.*

Proof. Let f_A and f_B be (Γ, w)-itineraries for A and B, respectively. Given a request set L contained in $A \cup B$, with $L = \{(s_1, t_1), \ldots, (s_\ell, t_\ell)\}$ and $\ell \leq (w+1) \cdot k$, and an integer $c' \in [c]$, we show how to correctly define $f_{A \cup B}(L, c')$ by looking at f_A, f_B, and the edges in D from A to B.

If L is contained in A we set $f_{A \cup B}(L, c') = f_A(L, c')$ as there are no edges from B to A in D, and set $f_{A \cup B}(L, c') = f_B(L, c')$ if L is contained in B. If there is a pair $(s, t) \in L$ such that $s \in B$ and $t \in A$, we set $f_{A \cup B}(L, c') = 0$. Assume now that no such pairs exist in L and that L is not contained in A nor in B.

Define $L_A = L_B = \emptyset$. For $i \in [\ell]$, do the following:

1. If $s_i \in A$ and $t_i \in A$, define $s_i^A = s_i$, $t_i^A = t_i$ and include the pair (s_i^A, t_i^A) in L_A.
2. If $s_i \in B$ and $t_i \in B$, define $s_i^B = s_i$, $t_i^B = t_i$ and include the pair (s_i^B, t_i^B) in L_B.
3. If $s_i \in A$ and $t_i \in B$, define $s_i^A = s_i$, $t_i^B = t_i$, choose $t_i^A \in A$ and $s_i^B \in B$ arbitrarily in such way that there is an edge from t_i^A to s_i^B in D, include (s_i^A, t_i^A) in L_A and (s_i^B, t_i^B) in L_B.

Figure 23 illustrates this construction. Now, for $c' \in [c]$, if $f_A(L_A, c_1) = f_B(L_B, c_2) = 1$ for some

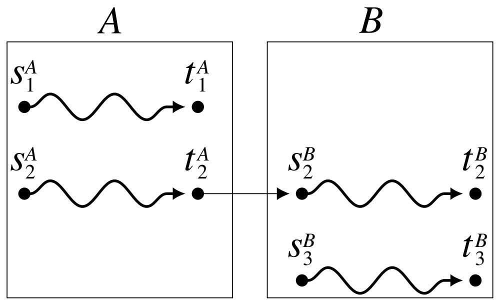


Figure 23 – Example of the construction from Lemma 4.3.8.

c_1, c_2 with $c_1 + c_2 \leq c'$, then we set $f_{A\cup B}(L, c') = 1$. Otherwise, we repeat the procedure used to construct L_A and L_B with a different choice for t_i^A and/or s_i^B in the third step. If all possible choices of L_A, L_B, c_1, and c_2 have been considered this way, we set $f_{A\cup B}(L, c') = 0$. We now show that this definition of $f_{A\cup B}(L, c')$ is correct.

Consider the instance $(D[A \cup B], L, \ell, c', s)$ of DEDP, let $k_1 = |L_A|$, and $k_2 = |L_B|$. If it is positive, then for some choice of L_A, L_B, c_1, and c_2, there are collections of paths $\mathcal{P}_A$ and $\mathcal{P}_B$ and integers c_1 and c_2 such that $\mathcal{P}_A$ and $\mathcal{P}_B$ are solutions for the instances $(D[A], L_A, k_1, c_1, s)$ and $(D[B], L_B, k_2, c_2, s)$ of DEDP, respectively. Thus, $f_A(L_A, c_1) = f_B(L_B, c_2) = 1$ since L_A and L_B are at most as large as L. Conversely, if the above equation holds for some choice of L_A, L_B, c_1, and c_2 such that $c_1 + c_2 \leq c'$, we can construct a solution for the instance on $D[A \cup B]$ by considering the union of a solution for $(D[A], L_A, k_1, c_1, s)$ with a solution for $(D[B], L_B, k_2, c_2, s)$, together with edges from targets of L_A to sources of L_B which where considered in step 3 described above. We conclude that the instance $(D[A \cup B], L, \ell, c', s)$ is positive if and only if there are integers c_1, c_2 and request sets L_A, L_B such that $c_1 + c_2 \leq c$ and

$$f_A(L_A, c_1) = f_B(L_B, c_2) = 1.$$

By definition, to compute a (Γ, w)-itinerary for $A \cup B$ we need to consider every request set of size at most $(w+1) \cdot k$ contained $A \cup B$. Thus there are $n^{2(w+1)\cdot k}$ choices of L. By construction, there are at most $n^{2(w+1\cdot k)}$ choices for L_A and L_B in total since we need to choose

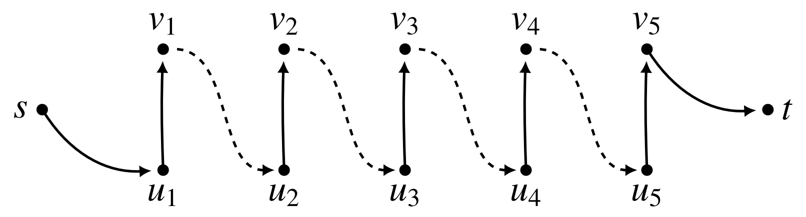


Figure 24 – Collections $\mathcal{P}_A$ and $\mathcal{P}_B$. A continuous line represents a piece of P contained in A and a dashed line represents a piece of P contained in B.

only the vertices t_i^A and s_i^B in step 3 of the construction of those requests sets. Finally, since $c' \in [c]$ and $c < n$, the bound on the running time follows. □

Lemma 4.3.9. *Let* Γ *be an instance of* DEDP *with* $\Gamma = (D, I, k, c, s)$ *and* $A, B \subseteq V(D)$ *such that* A *is* w*-guarded and* $|B| \leq w$*. Then a* (Γ, w)*-itinerary for* $A \cup B$ *can be computed from* (Γ, w)*-itineraries for* A *and* B *in time* $\mathcal{O}(n^{4(w+1)\cdot k+2})$*.*

Proof. Let f_A be a (Γ, w)-itinerary for A, L be a request set contained in $A \cup B$ with $L = \{(s_1, t_1), \ldots, (s_\ell, t_\ell)\}$ and $\ell \leq (w+1)\cdot k$, and $\Gamma_{c'}$ be the instance $(D[A \cup B], L, \ell, c', s)$ of DEDP, for $c' \in [c]$.

For each pair $(s,t) \in L$, a path from s to t in $D[A \cup B]$ in a solution for $\Gamma_{c'}$ may be entirely contained in A, entirely contained in B, or it may intersect both A and B. We can test if there is a solution for $\Gamma_{c'}$ whose paths are all contained in A by verifying the value of $f_A(L, c')$, and in time $\mathcal{O}(2^{w\cdot\ell})$ we can test if there is a solution for $\Gamma_{c'}$ that is entirely contained in B, since $|B| \leq w$. We now consider the case where all solutions for $\Gamma_{c'}$ contain a path intersecting both A and B.

Suppose that P is a path from s to t in a solution $\mathcal{P}$ for $\Gamma_{c'}$. Let $\mathcal{P}_A$ be the set of subpaths of P which are contained in A, with $\mathcal{P}_A = \{P_1^A, \ldots, P_a^A\}$, and, for $i \in [a]$, let u_i and v_i be the first and last vertices occurring in P_i^A, respectively. Furthermore, let $\mathcal{P}_B$ be the collection of subpaths of P contained in $B \cup (\bigcup_{i\in[a]}\{u_i, v_i\})$. Then $\mathcal{P}_B$ is a collection of disjoint paths satisfying the request set $\{(v_1, u_2), \ldots, (v_{a-1}, u_a)\}$, together with (s, u_1) if $s \in B$ and (v_a, t) if $t \in B$, such that each path of $\mathcal{P}_B$ has its extremities in $B \cup \{u_1, v_1, \ldots, u_a, v_a\}$ and all internal vertices in B. Figure 24 illustrates this case.

The number of such collections is a function depending on a and w only and, by Lemma 4.3.6 and our assumption that A is w-guarded, we can assume that $a \leq (w+1)\cdot k$. We show how we can test whether there is a solution for $\Gamma_{c'}$ using an itinerary for A and, for each $(s,t) \in L$, searching for a collection $\mathcal{P}_B$ as described above. Intuitively, we want to guess how the paths in a solution for $\Gamma_{c'}$ intersect A and how those pieces can be connected through B.

For $i \in [\ell]$, let $L_i = \{(u^i_1, v^i_1), \dots, (u^i_{\ell_i}, v^i_{\ell_i})\}$ and $L_A = L_1 \cup L_2 \cup \cdots \cup L_\ell$ (keeping each copy of duplicated entries) such that

1. $u^i_1 = s_i$ and $v^i_{\ell_i} = t_i$, for $i \in [\ell]$;
2. all vertices occuring in L_i are in A except possibly u^i_1 and $v^i_{\ell_i}$ (which may occur in $A \cup B$); and
3. $|L_A| \leq (w+1) \cdot k$.

By Lemma 4.3.6, we only need to consider request sets of size at most $(w+1) \cdot k$ in $A \cup B$ since every solution for Γ has at most $(w+1) \cdot k$ weak components in A. Let B^+ be the set formed by the union of B with all vertices occurring in L_A and in

$$L_B = \bigcup_{i \in [\ell]} \{(v^i_j, u^i_{j+1}) \mid j \in [\ell_i - 1]\}.$$

That is, for each pair $(u^i_j, v^i_j) \in L_A$ that we want to satisfy in A, we want to link this subpath in a (possible) solution for $\Gamma_{c'}$ to the next one through a path in B^+ satisfying the pair $(v^i_j, u^i_{j+1}) \in L_B$. We claim that there is a solution for $\Gamma_{c'}$ if and only if, for some choice of L_A, c_1, and L_B, we have $f_A(L_A, c_1) = 1$ and a collection of paths $\mathcal{P}_B$ satisfying L_B in $D[B^+]$ such that

(a) every path of $\mathcal{P}_B$ starts and ends in $B^+ \setminus B$ and has all of its internal vertices in B; and

(b) at most $c - c_1$ vertices of B occur in more than s paths of $\mathcal{P}_B$.

For the necessity, we can choose L_A and L_B as described above in this proof. For the sufficiency, let $\ell_A = \sum_{i \in [\ell]} \ell_i$ and $\mathcal{P}_A$ be a solution for the instance $(D[A], L_A, \ell_A, c_1, s)$ of DEDP, with $\mathcal{P}_A = \{P_1, \dots, P_{\ell_A}\}$. Now, since the paths in this collection are not necessarily disjoint, we are guaranteed to find only a directed walk from s_i to t_i for each pair $(s_i, t_i) \in L$ by linking (through the paths in B^+) the endpoints of the paths in the collection satisfying L_i, with $i \in [\ell]$. However, every such directed walk contains a path from s_i to t_i whose set of vertices is contained in the set of vertices of the walk. Thus by following those directed walks and choosing the paths appropriately, we can construct a solution for $\Gamma_{c'}$, since shortening the walks can only decrease the number of vertices occurring in $s+1$ or more paths of the collection.

The number of collections $\mathcal{P}_B$ for which (a) and (b) hold is $\mathcal{O}(2^{w \cdot \ell})$ and thus depending on k and w only, since $\ell \leq (w+1) \cdot k$. Since $|A| \leq n$, $c \leq n$, and the number of itineraries contained in $A \cup B$ is at most $n^{4(w+1) \cdot k}$, the bound on the running time follows. □

Finally, we obtain the XP algorithm combining Lemmas 4.3.8 and 4.3.9 together with Theorem 4.3.3.

Theorem 4.3.10. *The* DEDP *problem is solvable in time $\mathcal{O}(n^{4(w+1)\cdot k+3})$ in digraphs of directed tree-width at most w.*

4.3.2 Algorithms for the dual parameterization

We now show our algorithmic results for stronger parameterizations of DEDP including d as a parameter. The following definition is used in the description of the algorithms of this section.

Definition 4.3.11. *Let D be a graph, I be a request set with $I = \{(s_1,t_1),\dots,(s_k,t_k)\}$, and s be an integer. We say that a set $X \subseteq V(D)$ is* s-viable for *I if there is a collection of paths $\mathcal{P}$ satisfying I such that each vertex of X occurs in at most s paths of $\mathcal{P}$. We also say that $\mathcal{P}$ is* certifying X.

Thus an instance (D,I,k,c,s) of DEDP is positive if and only if D contains an s-viable set X with $|X| \geq d$. In other words, we want to find a set of vertices X of size at least d such that there is a collection of paths $\mathcal{P}$ satisfying I that is "well-behaved" inside of X; that is, the paths of $\mathcal{P}$ may intersect freely outside of X, but each vertex of X must be in at most s paths of $\mathcal{P}$. When $s = 1$, for instance, instead of asking for the paths to intersect only inside a small set of vertices (size at most c), we ask for them to be disjoint inside a large set of vertices (size at least d). Since we now consider d as a parameter instead of c, from this point onwards we may refer to instances of DEDP as (D,I,k,d,s).

Theorem 4.3.12. *There is an algorithm running in time $\mathcal{O}(n^{d+2}\cdot k^{d\cdot s})$ for the* DISJOINT ENOUGH DIRECTED PATHS *problem.*

Proof. Let D be a graph on n vertices and (D,I,k,d,s) be an instance of DEDP. Notice that if X is s-viable for I, then any proper subset of X is also s-viable for I. Therefore, we can restrict our attention to sets of size exactly d.

If $s = 0$, it is sufficient to test whether there is a d-sized set $X \subseteq V(D)$ such that there is a collection of paths satisfying I in $D \setminus X$, and this can be done in time $\mathcal{O}(n^d \cdot k \cdot n^2)$.

Let now that $s = 1$, and $I = \{(s_1,t_1),\dots,(s_k,t_k)\}$. We claim that a set $X \subseteq V(D)$ is 1-viable for I if and only if there is a partition $\mathcal{X}$ of X into sets $X_1,\dots,X_k$ such that $X \setminus X_i$ is not an (s_i,t_i)-separator, for $i \in [k]$.

Let $\mathcal{X}$ be as stated in the claim. For each $i \in [k]$, let P_i be a path from s_i to t_i in $D-(X \setminus X_i)$. Now, $\{P_1,\dots,P_k\}$ is a collection satisfying I and no pair of paths in it intersect inside X. Thus X is 1-viable for I as desired. For the necessity, since X is 1-viable for I, there is a collection of paths $P_1, \dots, P_k$ satisfying I such that $V(P_i) \cap V(P_j) \cap X = \emptyset$ for all $i, j \in [k]$ with $i \neq j$. Thus we choose $\mathcal{X} = \{X_1, \dots, X_k\}$ with $X_i = V(P_i) \cap X$ for $i \in [k-1]$ and $X_k = X \setminus (X_1, \dots, X_{k-1})$ and the claim follows.

Assume now that $X \subseteq V(D)$ with $|X| = d$. By the previous claim, we can check whether X is 1-viable for I by testing whether X admits a partition into (possibly empty) sets $X_1, \dots, X_k$ such that $X \setminus X_i$ is not an (s_i, t_i)-separator. Since $\mathsf{Stirling}(d, k) = \mathcal{O}(k^d)$, this yields an algorithm in time $\mathcal{O}(n^{d+2} \cdot k^d)$ for the DEDP problem when $s = 1$.

For $s \geq 2$, let $X = \{v_1, \dots, v_d\}$ and construct a graph D' from D by making s copies $v_i^1, \dots, v_i^s$ of each vertex $v_i \in X$ and adding one edge from each copy to each vertex in the neighborhood of v_i in D, respecting orientations.

For $i \in [d]$, let $V_i = \{v_i^1, \dots, v_i^s\}$ and $X' = \bigcup_{i \in [d]} V_i$. Now, there is a collection of paths $\mathcal{P}$ satisfying I in D such that each vertex in X is in at most s paths of $\mathcal{P}$ if and only if there is a collection of paths $\mathcal{P}'$ in D' such that no vertex in X' occurs in more than one path of $\mathcal{P}'$. To test whether a given X is s-viable for I with $s \geq 2$, we can just test whether X' is 1-viable for I in D'. Since $|X'| = d \cdot s$, this yields an algorithm in time $\mathcal{O}(n^{d+2} \cdot k^{d \cdot s})$ for DEDP. □

We now proceed to show that (k, d, s)-DEDP is FPT, by providing a kernel with at most $d \cdot 2^{k-s} \cdot \binom{k}{s} + 2k$ vertices. We start with some definitions and technical lemmas.

Notice that any vertex in D whose deletion disconnects more than s pairs in the request set I cannot be contained in any set X that is s-viable for I. Hence we make use of an operation to eliminate all such vertices from the input digraph while maintaining connectivity. We remind the reader that, for a request set I, we denote by $S(I)$ the set of source vertices in I and by $T(I)$ the set of target vertices in I (cf. Definition 2.4.2).

Definition 4.3.13 (Non-terminal vertices)**.** *Let (D, I, k, d, s) be an instance of* DEDP. *For a digraph D' such that $V(D') \subseteq V(D)$, we define $V^*(D') = V(D') \setminus (S(I) \cup T(I))$.*

That is, $V^*(D)$ is the set of non-terminal (i.e., neither source nor target) vertices of D.

Definition 4.3.14 (Congested vertex and blocking collection)**.** *Let (D, I, k, d, s) be an instance of* DEDP. *For $X \subseteq V^*(D)$, we define I_X as the subset of I that is* blocked *by X, that is, there are no paths from s to t in $D \setminus X$ for every $(s_i, t_i) \in I_X$. We say that a vertex $v \in V^*(D)$ is an (I, s)-*congested vertex *of D if $|I_{\{v\}}| \geq s + 1$. The* blocking collection *of I is the collection $\{B_1, \dots, B_k\}$ where $B_i = \{v \in V^*(D) \mid (s_i, t_i) \in I_{\{v\}}\}$, for $i \in [k]$. We say that D is* clean *for I and*

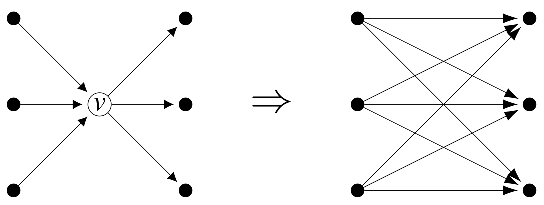


Figure 25 – Bypassing a vertex v.

that (D,I,k,d,s) *is a* clean instance *if there are no congested vertices in* $V^*(D)$*. When I and s are clear from the context, we drop them from the notation.*

We use the following operation to eliminate congested vertices of D while maintaining connectivity. It is used, for instance, in [79] (as the *torso* operation) and in [80].

Definition 4.3.15 (Bypassing vertices and sets)**.** *Let D be a graph and* $v \in V(D)$*. We refer to the following operation as* bypassing *v: delete v from D and, for each* $u \in N^-(v)$ *add one edge from u to each vertex* $w \in N^+(v)$*. We denote by* D/v *the graph generated by bypassing v in D. For a set of vertices* $B \subseteq V(D)$*, we denote by* D/B *the graph generated by bypassing, in D, all vertices of B in an arbitrary order.*

Figure 25 illustrates the bypass operation. We restrict our attention to vertices in $V^*(D)$ in Definition 4.3.14 because we want to avoid bypassing source or target vertices, and work only with vertices inside $V^*(D)$. Since $|S(I) \cup T(I)| \leq 2k$, we show later that this incurs an additive term of $2k$ in the size of the constructed kernel.

In [80, Lemma 3.6] the authors remark that the ending result of bypassing a set of vertices in a digraph does not depend on the order in which those vertices are bypassed. Furthermore, bypassing a vertex of D cannot generate a new congested vertex: if u is a congested vertex of D/v, then u is also a congested vertex of D, for any $v \in V(D) \setminus \{u\}$. Thus any instance (D,I,k,d,s) of DEDP is equivalent to the instance $(D/v,I,k,d,s)$, if v is a congested vertex of D, and arbitrarily bypassing a vertex of D can only make the problem harder. We formally state those observations below.

Lemma 4.3.16. *Let D be a digraph, I be a request set with* $I = \{(s_1,t_1),\ldots,(s_k,t_k)\}$*, s be an integer, B be the set of* (I,s)*-congested vertices of D, and* $D' = D/B$*. Then, with respect to I, X is s-viable in D if and only if X is s-viable in* D'*.*

Proof. Let X be an s-viable set for I in D. If $X \cap B \neq \emptyset$, then at least $s+1$ paths of any collection satisfying I must intersect in X, contradicting our choice for X, and hence $X \subseteq V(D')$. Similarly, if $X \subseteq V(D')$ then $X \cap B = \emptyset$ and the sufficiency follows. $\square$

Furthermore, from any solution for an instance resulting from bypassing a set of vertices in $V^*(D)$, we can construct a solution for the original instance in polynomial time by undoing the bypasses.

Remark 4.3.17. *Let (D,I,k,d,s) be an instance of* DEDP *and $Y \subseteq V^*(D)$. If $\mathcal{P}$ is a solution for $(D/Y,I,k,d,s)$, then (D,I,k,d,s) is positive and a solution can be constructed from $\mathcal{P}$ in polynomial time.*

The main ideas of the kernelization algorithm are the following. Let (D,I,k,d,s) be an instance of DEDP and $\{B_1,\dots,B_k\}$ be the blocking collection of I. First, we show that, if D is clean for I, there is an $i \in [k]$ such that $|V^*(D) \setminus B_i| \geq n/(k-s)k$ (Lemma 4.3.18). Then, we show that if D is clean and sufficiently large, and $|I| = s+1$, then the instance is positive and a solution can be found in polynomial time (Lemma 4.3.19).

Lemma 4.3.19 is used as the base case for our iterative algorithm. We start with the first instance, say (D,I,k,d,s), and proceed through $k-s+1$ iterations. At each iteration, we will choose one path from some s_i to its destination t_i such that a large part of the graph remains unused by any of the pairs chosen so far (by Lemma 4.3.18) and consider the request set containing only the remaining pairs for the next iteration. We repeat this procedure until we arrive at an instance where the number of requests is exactly $s+1$, and show that if n is large enough, then we can use Lemma 4.3.19 to output a solution for the last instance. From this solution, we extract a solution for (D,I,k,d,s) in polynomial time.

Lemma 4.3.18. *Let (D,I,k,d,s) be an instance of* DEDP, *$\{B_1,\dots,B_k\}$ be the blocking collection of I, and $n^* = |V^*(D)|$. If D is clean, then there is an $i \in [k]$ such that $|V^*(D/B_i)| \geq n^*(k-s)/k$ and there is a path P in D/B_i from s_i to t_i such that $|V^*(P)| \leq |V^*(D/B_i)|/2$.*

Proof. First, notice that

$$\sum_{v \in V^*(D)} |I_v| = \sum_{v \in V^*(D)} |\{B_i \mid v \in B_i\}| = \sum_{i \in [k]} |B_i|.$$

Now, if $|B_i| > n^* \cdot s/k$ for every $i \in [k]$, then there must be a vertex in v such that $|I_v| \geq s+1$, as in this case $\sum_{i \in [k]} |B_i| > n^* \cdot s$, contradicting our assumption that D is clean. We conclude that there is an $i \in [k]$ such that $|B_i| \leq n^* \cdot s/k$ and thus $V^*(D/B_i) = n^* - |B_i| \geq n^*(k-s)/k$, as desired.

The result trivially follows if there is a path P from s_i to t_i in D/B_i with $V^*(P) = \emptyset$. Thus we can assume that every path from s_i to t_i in D/B_i intersects $V^*(D/B_i)$ (see Figure 26). Let $X = (S(I) \cup T(I)) \setminus \{s_i,t_i\}$. By Menger's Theorem and since no vertex in $V^*(D/B_i)$ intersects

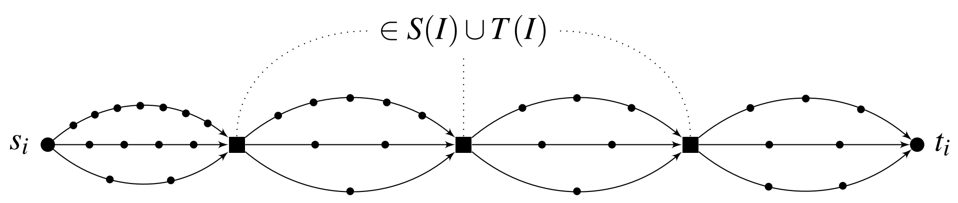


Figure 26 – Three paths from s_i to t_i in D/B_i. Square vertices are used to identify vertices in $S(I)\cup T(I)$, which may not be bypassed.

every path from s_i to t_i, there are two internally disjoint paths P_1 and P_2 from s_i to t_i in $(D/B_i)/X$. Without loss of generality, assume that P_1 is the shortest of those two paths, breaking ties arbitrarily. Then $|V^*(P_1)| \leq |V^*(D/B_i)|/2$ since P_1 and P_2 are disjoint, and the result follows. □

Figure 27 illustrates the procedure described in Lemma 4.3.18. We find a set B_i containing at most $n^* \cdot s/k$ vertices, and bypass all of its vertices in any order. Then we argue that a shortest path from s_i to t_i in D/B_i avoids a large set of vertices in D.

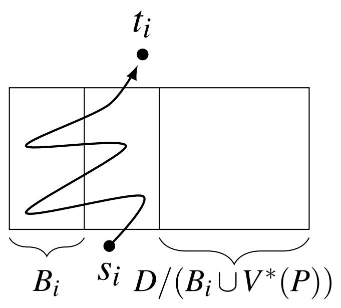


Figure 27 – A path P from s_i to t_i avoiding a large part of D.

Lemma 4.3.19. *Let (D,I,k,d,s) be an instance of* DEDP*, $m=|E(D)|$, and $n^*=V^*(D)$. If D is clean, $n^* \geq 2d(s+1)$, and $k=s+1$, then (D,I,k,d,s) is positive and a solution can be found in time $\mathcal{O}(k\cdot n(n+m))$.*

Proof. Let $\{B_1,\dots,B_k\}$ be the blocking collection of I and $D_i' = D/B_i$, for $i\in[k]$. By Lemma 4.3.18, there is an $i\in[k]$ such that $|V^*(D_i')| \geq n^*/(s+1)$ and a path P from s_i to t_i such that $V^*(P) \leq |V^*(D_i')|/2$. Let $D_i = D_i'/V^*(P)$. Now,

$$|V^*(D_i)| \;\geq\; \frac{|V^*(D_i')|}{2} \;\geq\; \frac{n^*}{2(s+1)}$$

and since $|I\setminus\{(s_i,t_i)\}| = s$, we are free to choose arbitrarily any collection of paths satisfying $I\setminus\{(s_i,t_i)\}$ in D_i. Reversing the bypasses done in D, this collection together with P_i yields a collection of paths satisfying I in D such that all vertices in $V^*(D_i)$ are contained in at most s of those paths. Since $n^* \geq 2d(s+1)$ by hypothesis, we have that $|V^*(D_i)| \geq d$ as required. We can generate the sets B_i in time $\mathcal{O}(n(n+m))$ by deleting a vertex of D and testing for connectivity between s_i and t_i. Thus a solution can be found in time $\mathcal{O}(k \cdot n(n+m))$, as desired. □

We are now ready to show the main ingredient of the algorithm: we provide a polynomial-time algorithm to solve large clean instances of the DISJOINT ENOUGH DIRECTED PATHS problem.

Theorem 4.3.20. *Let (D,I,k,d,s) be a clean instance of* DEDP *with $|V^*(D)| = n^* \geq d \cdot 2^{k-s} \cdot \binom{k}{s}$. Then (D,I,k,d,s) is positive and a solution can be found in time $\mathcal{O}(k \cdot n^2(n+m))$.*

Proof. Let $\mathcal{B}_0 = \{B_1,\dots,B_k\}$ be the blocking collection of I. We consider $\mathcal{B}_0$ to be sorted in non-decreasing order by the size of its elements and, by rearranging I if needed, we assume that this order agrees with I. For $i \in [k-(s+1)]$, we construct a sequence of sets $\{D_i, \mathcal{B}_i, \mathcal{P}_i\}$ where $n_i^* = |V^*(D_i)|$ and

(i) $\mathcal{B}_i = \{B_{i+1},\dots,B_k\}$;

(ii) $\mathcal{P}_i$ is a collection of paths $\{P_1,P_2,\dots,P_i\}$ such that P_j is a path from s_j to t_j in D_j, for $j \in [i]$; and

(iii) n_{i-1}^* is large enough to guarantee that we can find a path from s_i to t_i avoiding a large part of D_{i-1}. Formally, we want that

$$n_i^* \geq n_0^* \cdot \frac{(k-s)(k-s-1)\cdots(k-s-i+1)}{2^i \cdot k(k-1)\cdots(k-i+1)}.$$

We begin with $D_0 = D$, $n_0^* = n^*$, and $\mathcal{P}_0 = \emptyset$. Let $D_1' = D_0/B_1$. By applying Lemma 4.3.18 with input (D_0,I,k,d,s), we conclude that $|V^*(D_1')| \geq n^*(k-s)/k$ and there is a path P_1 from s_1 to t_1 in D_1' with $|V^*(P_1)| \leq |V^*(D_1')|/2$. Let $D_1 = D_1'/V^*(P_1)$ and $\mathcal{P}_1 = \{P_1\}$. Now,

$$n_1^* \geq \frac{|V^*(D_1')|}{2} \geq \frac{n_0^*(k-s)}{2k}$$

and conditions **(i)**, **(ii)**, and **(iii)** above hold for $(D_1,\mathcal{B}_1,\mathcal{P}_1)$. Assume that $i-1$ triples have been chosen in this way.

As before, we assume that $\mathcal{B}_{i-1}$ is sorted in non-increasing order by the size of its elements, and that this order agrees with $I \setminus I_{i-1}$. Furthermore, as D_0 is clean, so is D_{i-1}.

Let $D_i' = D_{i-1}/B_i$. Applying Lemma 4.3.18 with input $(D_{i-1}, I \setminus I_{i-1}, k-i+1, d, s)$, we conclude that $|V^*(D_i')| \geq n_i^*(k-i+1-s)/(k-i+1)$ and there is a path P_i from s_i to t_i in D_i'

with $|V^*(P_i)| \leq |V^*(D_i')|/2$. Let $\mathcal{P}_i = \mathcal{P}_{i-1} \cup \{P_i\}$ and $D_i = D_i'/B_i$. Then

$$n_i^* \;\geq\; n_{i-1}^* \cdot \frac{k-i+1-s}{2(k-i+1)}$$

and by our assumption that **(iii)** holds for n_{i-1} it follows that

$$\begin{aligned} n_i^* &\geq n_0^* \cdot \frac{(k-s)(k-s-1)\cdots(k-s-i+2)}{2^{i-1}k(k-1)\cdots(k-i+2)} \cdot \left(\frac{k-s-i+1}{2(k-i+1)}\right) \\ &= n_0^* \cdot \frac{(k-s)(k-s-1)\cdots(k-s-i+1)}{2^i \cdot k(k-1)\cdots(k-i+1)}, \end{aligned}$$

as desired and thus **(i)**, **(ii)**, and **(iii)** hold for $(D_i, \mathcal{B}_i, \mathcal{P}_i)$. The algorithm ends after iteration $k-(s+1)$. Following this procedure, we construct the collection $\mathcal{P}_{k-(s+1)} = \{P_1, P_2, \ldots, P_{k-(s+1)}\}$ satisfying **(ii)** and the graph $D_{k-(s+1)}$ with $n_{k-(s+i)}$ satisfying **(iii)**. Noticing that $|I \setminus I_{k-(s+1)}| = s+1$ (that is, only $s+1$ pairs in I are not accounted for in $\mathcal{P}_{k-(s+1)}$), it remains to show that our choice for n^* is large enough so that we are able to apply Lemma 4.3.19 on the instance $(D_{k-(s+1)}, I - I_{k-(s+1)}, s+1, d, s)$ of DEDP. That is, we want that $n_{k-(s+1)}^* \geq 2d(s+1)$. By **(iii)** it is enough to show that

$$n_{k-(s+1)}^* \;\geq\; n_0^* \cdot \frac{(k-s)(k-s-1)\cdots 3 \cdot 2}{2^{k-(s+1)} \cdot k(k-1)\cdots(s+3)(s+2)} \;\geq\; 2d \cdot (s+1),$$

and rewriting both sides of the fraction as $k!$ and $k!/(s+1)!$, respectively, we get

$$n_0 \cdot \frac{(k-s)!}{2^{k-(s+1)}} \;\geq\; 2d \cdot (s+1) \cdot \frac{k!}{(s+1)!} \;=\; \frac{2d \cdot k!}{s!},$$

which holds for

$$n_0 \;\geq\; \left(\frac{2^{k-(s+1)} \cdot 2d \cdot (s+1)}{(s+1)!}\right) \cdot \left(\frac{k!}{(k-s)!}\right) \;=\; d \cdot 2^{k-s} \cdot \binom{k}{s},$$

as desired.

Applying Lemma 4.3.19 with input $(D_{k-(s+1)}, I \setminus I_{k-(s+1)}, s+1, d, s)$ yields a collection $\hat{\mathcal{P}}$ satisfying $I \setminus I_{k-(s+1)}$ and a set $X \subseteq V(D)$ of size d such that X is disjoint from all paths in $\mathcal{P}_{k-(s+1)}$, since all vertices in $V^*(P)$ were bypassed in $D_{k-(s+1)}$ for every $P \in \mathcal{P}_{k-(s+1)}$, and all vertices in X occur in at most s paths of $\hat{\mathcal{P}}$. We can construct a collection of paths satisfying I from $\hat{\mathcal{P}} \cup \mathcal{P}_{k-(s+1)}$ by reversing all the bypasses done in D and connecting appropriately the paths in the collections (see Remark 4.3.17). We output this newly generated collection as a solution for (D, I, k, d, s).

For the running time, let $m = |(E(D)|$. We need time $\mathcal{O}(k \log k)$ to order the elements of $\mathcal{B}_0$, $\mathcal{O}(k \cdot n(n+m))$ to find the sets B_i, for $i \in [k]$, and $\mathcal{O}(n+m)$ to find each of the paths $\{P_1, \ldots, P_k\}$. Hence the algorithm runs in time $\mathcal{O}(k \cdot n^2(n+m))$. □

We acknowledge that it is possible to prove Theorem 4.3.20 without using Lemma 4.3.19 by stopping the iteration at the digraph D_{k-s} instead of D_{k-s-1}. However we believe it is easier to present the proof of Theorem 4.3.20 by having separate proofs for the iteration procedure (Lemma 4.3.18) that aims to generate an instance of DEDP for which we can apply our base case (Lemma 4.3.19).

Since any instance can be made clean in polynomial time, the kernelization algorithm for (k,d,s)-DEDP follows easily. Given an instance (D,I,k,d,s), we bypass all congested vertices of D to generate D'. If $|V^*(D')|$ is large enough to apply Theorem 4.3.20, the instance is positive and we can find a solution in polynomial time. Otherwise, we generated an equivalent instance (D',I,k,d,s) with $|V(D')|$ bounded from above by a function depending on k,d, and s only. As we restrict $|S(I)\cup T(I)| \leq 2k$, if D is clean and $V(D) \geq d\cdot 2^{k-s}\cdot\binom{k}{s}+2k$ we get the desired bound for $|V^*(D)|$. Thus, the following is a direct corollary of Theorem 4.3.20.

Theorem 4.3.21. *There is a kernelization algorithm running in time $\mathcal{O}(k\cdot n^2(n+m))$ that, given an instance (D,I,k,d,s) of* DEDP, *outputs either a solution for the instance or an equivalent instance (D',I,k,d,s) with $|V(D')| \leq d\cdot 2^{k-s}\cdot\binom{k}{s}+2k$.*

4.4 Concluding remarks

We introduced the DISJOINT ENOUGH DIRECTED PATHS problem and provided a number of hardness and algorithmic results, summarized in Table 1. Several questions remain open.

We showed that DEDP is NP-complete for every fixed $k \geq 3$ and $s \geq 1$. We do not know whether DEDP is also NP-complete for $k = 3$ and $s = 2$.

We provided an algorithm running in time $\mathcal{O}(n^{d+2}\cdot k^{d\cdot s})$ to solve the problem. This algorithm tests all partitions of a given $X \subseteq V(D)$ in search for one that respects some properties. Since there are at most $\binom{n}{d}$ subsets of $V(D)$ of size d, this yields an XP algorithm. The second term on the time complexity comes from the number of partitions of X we need to test. The problem may become easier if X is already given or, similarly, if d is a constant. In other words, is the (s)-DEDP problem FPT for fixed d?

Our main result is a kernel with at most $d\cdot 2^{k-s}\cdot\binom{k}{s}+2k$ vertices. The natural question is whether the problem admits a polynomial kernel with parameters k, d, and s, or even for fixed s. Notice that if there is a constant ℓ such that $k-s=\ell$, then the size of the kernel is $d \cdot 2^\ell \cdot k^\ell$, which is polynomial on d and k. The case $s = 0$ is also particularly interesting, as DEDP with $s = 0$ is equivalent to the STEINER NETWORK problem. In this case, we get a kernel of size at most $d \cdot 2^k + 2k$.

While we do not know whether (k,d,s)-DEDP admits a polynomial kernel, at least we are able to prove that a negative answer for $s = 0$ is enough to show that (k,d,s)-DEDP is unlikely to admit a polynomial kernel for any value of $s \geq 1$ when k is "far" from s, via the following polynomial time and parameter reduction.

Remark 4.4.1. *For any instance $(D,I,k,d,0)$ of* DEDP *and integer $s > 0$, one can construct in polynomial time an equivalent instance (D,I',k',d,s) of* DEDP *with $k' = k \cdot (d \cdot s + 1)$.*

Proof. For a request set I in D, let I' be the request set in D formed by $d \cdot s + 1$ copies of each pair in I and let $k' = k \cdot (d \cdot s + 1)$. We claim that an instance $(D,I,k,d,0)$ of DEDP is positive if and only if the associated instance (D,I',k',d,s), also of DEDP, is positive.

From any solution $\mathcal{P}$ for the first instance, we can construct a solution for the second by taking $d \cdot s + 1$ copies of each path in $\mathcal{P}$ and thus the necessity holds. For the sufficiency, let X be a s-viable set for (D,I',k',d,s) with certifying collection $\mathcal{P}'$. By the construction of I' and since at most $d \cdot s$ paths in $\mathcal{P}'$ can intersect X, we conclude that there is path $P \in \mathcal{P}'$ from s to t in $D \setminus X$ for each pair $(s,t) \in I$. Choosing all such paths we construct a collection $\mathcal{P}$ satisfying I in $D \setminus X$ and the result follows. □

In the undirected case, the STEINER TREE problem is unlikely to admit a polynomial kernel parameterized by k and c, with $c = n - d$ (in other words, the size of the solution); a simple proof for this result can be found in [6, Chapter 15]. Even if we consider a stronger parameter (that is, d instead of c), dealing with directed graphs may turn the problem much harder. We also remark that the problem admits a polynomial kernel in the undirected case if the input graph is planar [81]. It may also be the case for directed graphs.

REFERENCES

1 ROBERTSON, N.; SEYMOUR, P. D. Graph minors. XIII. The disjoint paths problem. **Journal of Combinatorial Theory, Series B**, Orlando, v. 63, n. 1, p. 65–110, 1995.

2 FORTUNE, S.; HOPCROFT, J.; WYLLIE, J. The directed subgraph homeomorphism problem. **Theoretical Computer Science**, Amsterdam, v. 10, n. 2, p. 111–121, 1980.

3 SLIVKINS, A. Parameterized tractability of edge-disjoint paths on directed acyclic graphs. **SIAM Journal on Discrete Mathematics**, Philadelphia, v. 24, n. 1, p. 146–157, 2010.

4 KAWARABAYASHI, K.-i.; KREUTZER, S. The Directed Grid Theorem. *In*: ACM SYMPOSIUM ON THEORY OF COMPUTING, 47., 2015, Portland. **Proceedings** ... New York: Association for Computing Machinery, 2015. p. 655–664.

5 ROBERTSON, N.; SEYMOUR, P. D. Graph minors. V. Excluding a planar graph. **Journal of Combinatorial Theory, Series B**, Orlando, v. 41, n. 01, p. 92–114, 1986.

6 CYGAN, M.; FOMIN, F. V.; KOWALIK, L.; LOKSHTANOV, D.; MARX, D.; PILIPCZUK, M.; PILIPCZUK, M.; SAURABH, S. **Parameterized algorithms**. London: Springer, 2015.

7 KARP, R. M. Reducibility among combinatorial problems. *In*: MILLER, R. E.; THATCHER, J. W. (orgs.). **Complexity of computer computations**. New York: Springer US, 1972. p. 85–103.

8 DOWNEY, R. G.; FELLOWS, M. R. Fixed-parameter tractability and completeness ii: On completeness for W[1]. **Theoretical Computer Science**, Orlando, v. 141, n. 1, p. 109 – 131, 1995.

9 DREYFUS, S. E.; WAGNER, R. A. The steiner problem in graphs. **Networks**, Hoboken, v. 1, n. 3, p. 195–207, 1971.

10 BOUSQUET, N.; DALIGAULT, J.; THOMASSÉ, S. Multicut is FPT. **SIAM Journal on Computing**, Philadelphia, v. 47, n. 1, p. 166–207, 2018.

11 MARX, D.; RAZGON, I. Fixed-parameter tractability of multicut parameterized by the size of the cutset. **SIAM Journal on Computing**, Philadelphia, v. 43, n. 2, p. 355–388, 2014.

12 GUO, J.; NIEDERMEIER, R.; SUCHÝ, O. Parameterized complexity of arc-weighted directed steiner problems. **SIAM Journal on Discrete Mathematics**, Philadelphia, v. 25, n. 2, p. 583–599, 2011.

13 PILIPCZUK, M.; WAHLSTRÖM, M. Directed Multicut is W[1]-hard, even for four terminal pairs. **ACM Transactions on Computation Theory**, New York, v. 10, n. 3, p. 13:1–13:18, 2018.

14 FELDMANN, A. E.; MARX, D. The complexity landscape of fixed-parameter directed steiner network problems. *In*: INTERNATIONAL COLLOQUIUM ON AUTOMATA, LANGUAGES, AND PROGRAMMING, 43., 2016, Orlando. **Proceedings** ... Dagstuhl: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, v. 53, 2016. p. 27:1–27:14.

15 BERTELE, U.; BRIOSCHI, F. **Nonserial dynamic programming**. Orlando: Academic Press Inc., 1972.

16 HALIN, R. *S*-functions for graphs. **Journal of Geometry**, London, v. 8, n. 1, p. 171–186, 1976.

17 CAMPOS, V.; LOPES, R.; MAIA, A. K.; SAU, I. Adapting the Directed Grid Theorem into an FPT algorithm. *In*: LATIN AND AMERICAN ALGORITHMS, GRAPHS AND OPTIMIZATION SYMPOSIUM, 10., 2019, Belo Horizonte. **Proceedings** ... Belo Horizonte: Electronic Notes in Theoretical Computer Science, v. 346, 2019. p. 229–240.

18 CHEKURI, C.; CHUZHOY, J. Polynomial bounds for the grid-minor theorem. **Journal of the ACM**, New York, v. 63, n. 5, p. 40:1–40:65, 2016.

19 CHUZHOY, J.; TAN, Z. Towards tight(er) bounds for the Excluded Grid Theorem. *In*: SYMPOSIUM ON DISCRETE ALGORITHMS, 30., 2019, San Diego. **Proceedings** ... Philadelphia: Society for Industrial and Applied Mathematics, 2019. p. 1445–1464.

20 FOMIN, F. V.; DEMAINE, E. D.; HAJIAGHAYI, M. T.; THILIKOS, D. M. Bidimensionality. *In*: KAO, M. (org.) **Encyclopedia of algorithms**. New York: Springer US, 2016. p. 203–207.

21 DEMAINE, D.; FOMIN, V.; HAJIAGHAYI, M.; THILIKOS, D. M. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. **Journal of the ACM**, New York, v. 52, n. 6, p. 866–893, 2005.

22 ROBERTSON, N.; SEYMOUR, P. D. Graph minors. XXI. Graphs with unique linkages. **Journal of Combinatorial Theory, Series B**, Orlando, v. 99, n. 3, p. 583–616, 2009.

23 ROBERTSON, N.; SEYMOUR, P. D. Graph Minors. XXII. Irrelevant vertices in linkage problems. **Journal of Combinatorial Theory, Series B**, Orlando, v. 102, n. 2, p. 530–563, 2012.

24 JOHNSON, T.; ROBERTSON, N.; SEYMOUR, P. D.; THOMAS, R. Directed tree-width. **Journal of Combinatorial Theory, Series B**, Orlando, v. 82, n. 01, p. 138–154, 2001.

25 LOPES, R.; SAU, I. A relaxation of the Directed Disjoint Paths problem: a global congestion metric helps. *In*: INTERNATIONAL SYMPOSIUM ON MATHEMATICAL FOUNDATIONS OF COMPUTER SCIENCE, 45., 2020, Prague. **Proceedings** ... Dagstuhl: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, v. 170, 2020. p. 66:1–66:15.

26 LYNCH, J. F. The equivalence of theorem proving and the interconnection problem. **ACM SIGDA Newsletter**, New York, v. 5, n. 3, p. 31–36, 1975.

27 AMIRI, S. A.; KREUTZER, S.; MARX, D.; RABINOVICH, R. Routing with congestion in acyclic digraphs. **Information Processing Letters**, Amsterdam, v. 151, 2019.

28 SCHRIJVER, A. Finding k disjoint paths in a directed planar graph. **SIAM Journal on Computing**, Philadelphia, v. 23, n. 4, p. 780–788, 1994.

29 CYGAN, M.; MARX, D.; PILIPCZUK, M.; PILIPCZUK, M. The planar Directed k-Vertex-Disjoint Paths problem is fixed-parameter tractable. *In*: IEEE SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE, 54., 2013, Berkley. **Proceedings** ... Washington: IEEE Computer Society, 2013. p. 197–206.

30 BONDY, A.; MURTY, M. R. **Graph theory**. London: Springer-Verlag, 2008.

31 MENGER, K. Zur allgemeinen kurventheorie. **Fundamenta mathematicae**, Warszawa, v. 10, n. 1, p. 96–115, 1927.

32 BOYADZHIEV, K. N. Close encounters with the Stirling Numbers of the second kind. **Mathematics Magazine**, Washington, v. 85, n. 4, p. 252–266, 2012.

33 DOWNEY, R. G.; FELLOWS, M. R. **Parameterized complexity**. New York: Springer-Verlag, 1999.

34 DOWNEY, R. G.; FELLOWS, M. R. **Fundamentals of parameterized complexity**. London: Springer-Verlag, 2013.

35 ARNBORG, S.; CORNEIL, D. G.; PROSKUROWSKI, A. Complexity of finding embeddings in a k-tree. **SIAM Journal on Algebraic Discrete Methods**, Philadelphia, v. 8, n. 2, p. 277–284, 1987.

36 BODLAENDER, H. L. A linear-time algorithm for finding tree-decompositions of small treewidth. **SIAM Journal on Computing**, Philadelphia, v. 25, n. 6, p. 1305–1317, 1996.

37 ROBERTSON, N.; SEYMOUR, P. Graph minors. II. algorithmic aspects of tree-width. **Journal of Algorithms**, Orlando, v. 7, n. 3, p. 309–322, 1986.

38 COURCELLE, B. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. **Information and Computation**, Amsterdam, v. 85, n. 1, p. 12–75, 1990.

39 ROBERTSON, N.; SEYMOUR, P.; THOMAS, R. Quickly excluding a planar graph. **Journal of Combinatorial Theory, Series B**, Orlando, v. 62, n. 2, p. 323–348, 1994.

40 GU, Q.-P.; TAMAKI, H. Improved bounds on the planar branchwidth with respect to the largest grid minor size. **Algorithmica**, London, v. 64, n. 3, p. 416–453, 2012.

41 REED, B. Introducing directed tree-width. **Electronic Notes in Discrete Mathematics**, Amsterdam, v. 3, p. 222–229, 1999.

42 OLIVEIRA, M. O. An algorithmic metatheorem for directed treewidth. **Discrete Applied Mathematics**, Amsterdam, v. 204, p. 49–76, 2016.

43 BODLAENDER, H. L. Treewidth: characterizations, applications, and computations. *In*: INTERNATIONAL WORKSHOP ON GRAPH-THEORETIC CONCEPTS IN COMPUTER SCIENCE, 32., 2006, Bergen. **Proceedings** ... London: Springer, 2006. p. 1–14.

44 KOSTER, A.; HOESEL, S. van; KOLEN, A. Solving frequency assignment problems via tree-decomposition. **Electronic Notes in Discrete Mathematics**, Amsterdam, v. 3, p. 102–105, 1999.

45 COOK, W.; SEYMOUR, P. D. Tour merging via branch-decompositions. **INFORMS Journal on Computing**, Catonsville, v. 15, p. 233–248, 2003.

46 GROHE, M.; KAWARABAYASHI, K.-i.; MARX, D.; WOLLAN, P. Finding topological subgraphs is fixed-parameter tractable. *In*: ACM SYMPOSIUM ON THEORY OF COMPUTING, 43., 2011, San Jose. **Proceedings** ... New York: Association for Computing Machinery, 2011. p. 479–488.

47 KLEINBERG, J. M. Decision algorithms for unsplittable flow and the half-disjoint paths problem. *In*: ACM SYMPOSIUM ON THEORY OF COMPUTING, 30., 1998, Dallas. **Proceedings** ... New York: Association for Computing Machinery, 1998. p. 530–539.

48 KAWARABAYASHI, K.; KREUTZER, S. An excluded grid theorem for digraphs with forbidden minors. *In*: ACM-SIAM SYMPOSIUM ON DISCRETE ALGORITHMS, 25., 2014, Portland. **Proceedings** ... Philadelphia: Society for Industrial and Applied Mathematics, 2014. p. 72–81.

49 JOHNSON, T.; ROBERTSON, N.; SEYMOUR, P. D.; THOMAS, R. Excluding a grid minor in planar digraphs. Preprint in the computing research repository. 2015. Available at: https://arxiv.org/abs/1510.00473.

50 HATZEL, M.; KAWARABAYASHI, K.; KREUTZER, S. Polynomial planar Directed Grid Theorem. *In*: ACM-SIAM SYMPOSIUM ON DISCRETE ALGORITHMS, 30., 2019, San Diego. **Proceedings** ... Philadelphia: Society for Industrial and Applied Mathematics, 2019. p. 1465–1484.

51 AMIRI, S. A.; KAWARABAYASHI, K.; KREUTZER, S.; WOLLAN, P. The Erdős-Pósa property for directed graphs. Preprint in the computing research repository. 2016. Available at: https://arxiv.org/abs/1603.02504.

52 EDWARDS, K.; MUZI, I.; WOLLAN, P. Half-integral linkages in highly connected directed graphs. *In*: EUROPEAN SYMPOSIUM ON ALGORITHMS, 25., 2017, Vienna. **Proceedings** ... Dagstuhl: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, v. 87, 2017. p. 36:1–36:12.

53 CHEKURI, C.; ENE, A.; PILIPCZUK, M. Constant congestion routing of symmetric demands in planar directed graphs. *In*: INTERNATIONAL COLLOQUIUM ON AUTOMATA, LANGUAGES, AND PROGRAMMING, 43., 2016, Rome. **Proceedings** ... Dagstuhl: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, v.55, 2016. p. 7:1–7:14.

54 BANG-JENSEN, J.; GREGORY, G. **Classes of directed graphs**. London: Springer, 2018.

55 FLUM, J.; GROHE, M. **Parameterized Complexity Theory**. London: Springer, 2006.

56 ERBACHER, R. F.; JAEGER, T.; TALELE, N.; TEUTSCH, J. Directed multicut with linearly ordered terminals. Preprint in the computing research repository. 2014. Available at: https://arxiv.org/abs/1407.7498.

57 SEYMOUR, P. D.; THOMAS, R. Graph searching and a min-max theorem for tree-width. **Journal of Combinatorial Theory, Series B**, Orlando, v. 58, n. 1, p. 22–33, 1993.

58 MATTHIAS, D.; EMMERT-STREIB, F. **Quantitative Graph Theory:** Mathematical Foundations and Applications. Boca Raton: CRC Press, 2014.

59 KRATSCH, S.; PILIPCZUK, M.; PILIPCZUK, M.; WAHLSTRÖM, M. Fixed-parameter tractability of multicut in directed acyclic graphs. **SIAM Journal on Discrete Mathematics**, Philadelphia, v. 29, n. 1, p. 122–144, 2015.

60 CHEN, J.; LIU, Y.; LU, S.; O'SULLIVAN, B.; RAZGON, I. A fixed-parameter algorithm for the directed feedback vertex set problem. **Journal of the ACM**, New York, v. 55, n. 5, p. 21:1–21:19, 2008.

61 CAYLEY, A. On the analytical forms called trees. **American Journal of Mathematics**, Baltimore, v. 4, n. 1, p. 266–268, 1881.

62 BELL, E. T. Exponential polynomials. **Annals of Mathematics**, Princeton, v. 35, n. 2, p. 258–277, 1934.

63 KREUTZER, S.; ORDYNIAK, S. Digraph decompositions and monotonicity in digraph searching. **Theoretical Computer Science**, Amsterdam, v. 412, n. 35, p. 4688–4703, 2011.

64 GANIAN, R.; HLINĚNÝ, P.; KNEIS, J.; LANGER, A.; OBDRŽÁLEK, J.; ROSSMANITH, P. Digraph width measures in parameterized algorithmics. **Discrete Applied Mathematics**, Amsterdam, v. 168, p. 88–107, 2014.

65 BEZÁKOVÁ, I.; CURTICAPEAN, R.; DELL, H.; FOMIN, F. Finding detours is fixed-parameter tractable. **SIAM Journal on Discrete Mathematics**, Philadelphia, v. 33, n. 4, p. 2326–2345, 2016.

66 BONAMY, M.; KOWALIK, L.; NEDERLOF, J.; PILIPCZUK, M.; SOCALA, A.; WROCHNA, M. On directed feedback vertex set parameterized by treewidth. *In*: INTERNATIONAL WORKSHOP ON GRAPH-THEORETIC CONCEPTS IN COMPUTER SCIENCE, 44., 2018, Cottbus. **Proceedings** ... London: Springer, 2018. p. 65–78.

67 BERGOUGNOUX, B.; EIBEN, E.; GANIAN, R.; ORDYNIAK, S.; RAMANUJAN, M. S. Towards a polynomial kernel for Directed Feedback Vertex Set. **Algorithmica**, London, v. 83, n. 5, p. 1201–1221, 2021.

68 GIANNOPOULOU, A. C.; KAWARABAYASHI, K.; KREUTZER, S.; KWON, O. The Directed Flat Wall Theorem. *In*: ACM-SIAM SYMPOSIUM ON DISCRETE ALGORITHMS, 13., 2020, Salt Lake City. **Proceedings** ... Philadelphia: Society for Industrial and Applied Mathematics, 2020. p. 239–258.

69 DORN, F.; FOMIN, F. V.; LOKSHTANOV, D.; RAMAN, V.; SAURABH, S. Beyond bidimensionality: parameterized subexponential algorithms on directed graphs. **Information and Computation**, Amsterdam, v. 233, p. 60–70, 2013.

70 THOMASSEN, C. Highly connected non-2-linked digraphs. **Combinatorica**, London, v. 11, n. 4, p. 393–395, 1991.

71 EDWARDS, K.; MUZI, I.; WOLLAN, P. Half-integral linkages in highly connected directed graphs. *In*: **European Symposium on Algorithms (ESA), 25., 2017**. [*S. l.: s. n.*]. p. 36:1–36:12.

72 KAWARABAYASHI, K.; KOBAYASHI, Y.; KREUTZER, S. An excluded half-integral grid theorem for digraphs and the Directed Disjoint Paths problem. *In*: ACM SYMPOSIUM ON THEORY OF COMPUTING, 46., 2014, San Jose. **Proceedings** ... New York: Association for Computing Machinery, 2014. p. 70–78.

73 JONES, M.; LOKSHTANOV, D.; RAMANUJAN, M. S.; SAURABH, S.; SUCHÝ, O. Parameterized complexity of Directed Steiner Tree on sparse graphs. **SIAM Journal on Discrete Mathematics**, Philadelphia, v. 31, n. 2, p. 1294–1327, 2017.

74 MÖLLE, D.; RICHTER, S.; ROSSMANITH, P. Enumerate and expand: improved algorithms for connected vertex cover and tree cover. **Theory of Computing Systems**, London, v. 43, n. 2, p. 234–253, 2008.

75 CHOR, B.; FELLOWS, M.; JUEDES, D. W. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. *In*: INTERNATIONAL WORKSHOP ON GRAPH-THEORETIC CONCEPTS IN COMPUTER SCIENCE, 30., 2004, Bad Honnef. **Proceedings** ... London: Springer, 2004. p. 257–269.

76 BASAVARAJU, M.; FRANCIS, M. C.; RAMANUJAN, M. S.; SAURABH, S. Partially polynomial kernels for Set Cover and Test Cover. **SIAM Journal on Discrete Mathematics**, Philadelphia, v. 30, n. 3, p. 1401–1423, 2016.

77 DUH, R.; FÜRER, M. Approximation of k-Set Cover by semi-local optimization. *In*: ACM SYMPOSIUM ON THE THEORY OF COMPUTING, 29., 1997, El Paso. **Proceedings** ... New York: Association for Computing Machinery, 1997. p. 256–264.

78 ARAÚJO, J.; CAMPOS, V. A.; LIMA, C. V. G. C.; SANTOS, V. F.; SAU, I.; SILVA, A. Dual parameterization of Weighted Coloring. **Algorithmica**, London, v. 82, n. 8, p. 2316–2336.

79 CHITNIS, R.; HAJIAGHAYI, M.; MARX, D. Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. **SIAM Journal on Computing**, Philadelphia, v. 42, n. 4, p. 1674–1696, 2013.

80 KRATSCH, S.; PILIPCZUK, M.; PILIPCZUK, M.; WAHLSTRöM, M. Fixed-parameter tractability of multicut in directed acyclic graphs. **SIAM Journal on Discrete Mathematics**, Philadelphia, v. 29, n. 1, p. 122–144, 2015.

81 PILIPCZUK, M.; PILIPCZUK, M.; SANKOWSKI, P.; LEEUWEN, E. J. V. Network sparsification for steiner problems on planar and bounded-genus graphs. **ACM Transactions on Algorithms**, New York, v. 14, n. 4, p. 53:1–53:73, 2018.

APPENDIX A – LIST OF DEFINITIONS

Arboreal Decomposition: An *arboreal decomposition* β of a digraph D is a triple $(R, \mathcal{X}, \mathcal{W})$ where R is an arborescence, $\mathcal{X} = \{X_e : e \in E(R)\}$, $\mathcal{W} = \{W_r : r \in V(R)\}$, and $\mathcal{X}, \mathcal{W}$ are collections of sets of vertices of D (called *bags*) such that

(i) $\mathcal{W}$ is a partition of $V(D)$ into non-empty sets, and

(ii) if $e \in E(R)$, then $\bigcup\{W_r : r \in V(R) \text{ and } r > e\}$ is X_e-guarded.

We also say that r is a *leaf* of $(R, \mathcal{X}, \mathcal{W})$ if r has out-degree zero in R.

Brambles in digraphs: A *bramble* $\mathcal{B} = \{B_1, \ldots, B_\ell\}$ in a digraph D is a family of strongly connected subgraphs of D such that if $\{B, B'\} \subseteq \mathcal{B}$ then $V(B) \cap V(B') \neq \emptyset$ or there are edges in D from $V(B)$ to $V(B')$ and from $V(B')$ to $V(B)$. A *hitting set* of a bramble $\mathcal{B}$ is a set $C \subseteq V(D)$ such that $C \cap V(B) \neq \emptyset$ for all $B \in \mathcal{B}$. The *order* of a bramble $\mathcal{B}$, denoted by $\mathsf{ord}(\mathcal{B})$, is the minimum size of a hitting set of $\mathcal{B}$. The *bramble number* of a digraph D, denoted by $\mathsf{bn}(D)$, is the the maximum k such that D admits a bramble of order k.

Bypassing vertices and sets: Let D be a graph and $v \in V(D)$. We refer to the following operation as *bypassing* v: delete v from D and, for each $u \in N^-(v)$ add one edge from u to each vertex $w \in N^+(v)$. We denote by D/v the graph generated by bypassing v in D. For a set of vertices $B \subseteq V(D)$, we denote by D/B the graph generated by bypassing, in D, all vertices of B in an arbitrary order.

Butterfly contraction and butterfly minors: Let D be a digraph. An edge e from u to v of D is *butterfly contractible* if e is the only outgoing edge of u or the only incoming edge of v. By *butterfly contracting* e in D, we obtain a digraph D' with vertex set $V(D') = V(D) \setminus \{u, v\} \cup x_{u,v}$, where $x_{u,v}$ is a new vertex, and $E(D') = E(D) \setminus \{e\}$. Every incidence of an edge $f \in E(D')$ to u or v in D becomes an incidence to $x_{u,v}$ in D'. If D' is generated from a subgraph of D by a series a butterfly contractions, we say that D' is a *butterfly minor* of D.

Congested vertex and blocking collection: Let (D, I, k, d, s) be an instance of DEDP. For $X \subseteq V^*(D)$, we define I_X as the subset of I that is *blocked* by X, that is, there are no paths from s to t in $D \setminus X$ for every $(s_i, t_i) \in I_X$. We say that a vertex $v \in V^*(D)$ is an (I, s)-*congested vertex* of D if $|I_{\{v\}}| \geq s + 1$. The *blocking collection of* I is the collection $\{B_1, \ldots, B_k\}$ where $B_i = \{v \in V^*(D) \mid (s_i, t_i) \in I_{\{v\}}\}$, for $i \in [k]$. We say that D is *clean for* I and that (D, I, k, d, s) is a *clean instance* if there are no congested vertices in $V^*(D)$. When I and s are clear from the context, we drop them from the notation.

Cylindrical grid: A *cylindrical grid of order k* is a digraph formed by the union of k disjoint cycles $C_1, \ldots, C_k$ and $2k$ disjoint paths P_1, P_2, …, P_{2k} where

1. for $i \in [k], V(C_i) = \{v_{i,1}, v_{i,2}, \ldots, v_{i,2k}\}$ and $E(C_i) = \{(v_{i,j}, v_{i,j+1} \mid j \in [2k-1])\} \cup \{(v_{i,2k}, v_{i,1})\}$,
2. for $i \in \{1,3,\ldots,2k-1\}$, $E(P_i) = \{(v_{1,i}, v_{2,i}), (v_{2,i}, v_{3,i}), \ldots, (v_{k-1,i}, v_{k,i})\}$, and
3. for $i \in \{2,4,\ldots,2k\}$, $E(P_i) = \{(v_{k,i}, v_{k-1,i}), (v_{k-1,i}, v_{k-2,i}), \ldots, (v_{2,i}, v_{1,i})\}$.

Directed tree-width: Let $(R, \mathcal{X}, \mathcal{W})$ be an arboreal decomposition of a digraph D. For a vertex $r \in V(R)$, we denote by $\mathsf{width}(r)$ the size of the set $W_r \cup (\bigcup_{e \sim r} X_e)$. The *width* of $(R, \mathcal{X}, \mathcal{W})$ is the least integer k such that, for all $r \in V(R)$, $\mathsf{width}(r) \leq k+1$. The *directed tree-width* of D, denoted by $\mathsf{dtw}(D)$, is the least integer k such that D has an arboreal decomposition of width k.

Havens in digraphs: Let D be a digraph. A *haven of order k* in D is a function β assigning to every set $Z \subseteq V(D)$, with $|Z| \leq k-1$, the vertex set of a strong component of $D \setminus Z$ in such way that if $Z' \subseteq Z \subseteq V(D)$ then $\beta(Z) \subseteq \beta(Z')$. The *haven number* of a digraph D, denoted by $\mathsf{hn}(D)$, is the maximum k such that D admits a haven of order k.

(i)-splits: An *(i)-split* $\mathcal{S}$ of P is a collection formed by a set $\{Q_j \mid j \in [i]\}$ of subpaths of P, a subpath P_i of P, a set of brambles $\{\mathcal{B}_j \mid j \in [i]\}$, a set of vertices $\{a_j \mid j \in [i]\}$, and a set of vertices X_i such that

1. for $j \in [i]$, vertex a_j is the sucessor in P of the last vertex of Q_j, and, if $j \leq i-1$, the first vertex of Q_{j+1} is the sucessor in P of vertex a_j,
2. for $j \in [i]$, $\mathsf{ord}(\mathcal{B}_j) \geq \lfloor k/2 \rfloor$,
3. for $j \in [i]$, $\mathcal{B}_j \subseteq \mathcal{B}_T$ and $V(Q_j)$ is a hitting set of $\mathcal{B}_j$,
4. P_i is the subpath of P from the sucessor in P of the last vertex of Q_i to the last vertex of P, and
5. $X_i = \bigcup_{j \in [i]} (V(P_j) \cup \{a_j\})$, and

$$\mathsf{ord}(\overline{B}(X_i)) \geq g(k) - i\left(\left\lfloor \frac{k}{2} \right\rfloor + 1\right).$$

Itinerary: Let Γ be an instance of DEDP with $\Gamma = (D, I, k, c, s)$, $A \subseteq V(D)$, and $\mathcal{I}_A$ be the set of all request sets on D which are contained in A. For an integer w, a (Γ, w)*-itinerary for* A is a function $f_A : \mathcal{I}_A \times \mathbb{N} \to \{0,1\}$ such that $f_A(I', c') = 1$ if and only if

(i) $k' \leq (w+1) \cdot k$, for $k' = |I'|$;

(ii) $c' \leq c$; and

(iii) the instance $(D[A], I', k', c', s)$ of DEDP is positive.

Limited collections: Let I be a request set in a digraph D with $|I| = k$ and $\mathcal{P}$ be a collection of paths satisfying I. We say that $\mathcal{P}$ is (k, w, S)*-limited*, for some $S \subseteq V(D)$, if $D(\mathcal{P}) \subseteq D[S]$ and for every w-guarded set $S' \subseteq S$, the digraph induced by $V(D(\mathcal{P})) \cap S'$ has at most $(w+1) \cdot k$ weak components.

Non-terminal vertices: Let (D, I, k, d, s) be an instance of DEDP. For a digraph D' such that $V(D') \subseteq V(D)$, we define $V^*(D') = V(D') \setminus (S(I) \cup T(I))$.

(T, r)**-Balanced separators and** (k, r)**-linked sets**: Let D be a digraph, $T \subseteq V(D)$, and r be a non-negative integer. We say that a set $Z \subseteq V(D)$ is a (T, r)-balanced separator if for every strong component C of $D \setminus Z$, we have $|T \cap V(C)| \leq r$. If the minimum size of a (T, r)-balanced separator is at least $k+1$, we say that T is (k, r)-linked.

Requests and satisfying collections: Let D be a digraph and $\mathcal{P}$ be a collection of paths of D. A *request* in D is a pair of vertices of D. For a request set $I = \{(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)\}$, we say that the vertices $\{s_1, s_2, \ldots, s_k\}$ are *source* vertices and that $\{t_1, t_2, \ldots, t_k\}$ are *target* vertices, and we refer to them as $S(I)$ and $T(I)$, respectively. We say that $\mathcal{P}$ *satisfies* I if $\mathcal{P} = \{P_1, \ldots, P_k\}$ and P_i is a path from s_i to t_i, for $i \in [k]$.

T**-brambles**: Let D be a digraph and $T \subseteq V(D)$ with $|T| = 2k - 1$. The T*-bramble* $\mathcal{B}_T$ of D is defined as $\mathcal{B}_T = \{B \subseteq D \mid B \text{ is induced, strongly connected, and } |V(B) \cap T| \geq k\}$.

s**-viable sets**: Let D be a graph, I be a request set with $I = \{(s_1, t_1), \ldots, (s_k, t_k)\}$, and s be an integer. We say that a set $X \subseteq V(D)$ is s*-viable for* I if there is a collection of paths $\mathcal{P}$ satisfying I such that each vertex of X occurs in at most s paths of $\mathcal{P}$. We also say that $\mathcal{P}$ is *certifying* X.

Well-linked sets: Let D be a digraph and $A \subseteq V(D)$. We say that A is *well-linked* in D if, for all disjoint $X, Y \subseteq A$ with $|X| = |Y|$, there are $|X|$ vertex-disjoint paths from X to Y in D. The *order* of a well-linked set A is $|A|$. We denote by $\mathsf{wlink}(D)$ the size of a largest well-linked set in D.

Z**-guarded sets**: Let D be a digraph, $Z \subseteq V(D)$, and $S \subseteq V(D) \setminus Z$. We say that S is Z*-guarded* if there is no directed walk in $D \setminus Z$ with first and last vertices in S that uses a vertex of $D \setminus (Z \cup S)$.

APPENDIX B – COLLECTION OF OTHER WORKS

We include abstracts of other works obtained during the Ph.D.

1. An extended abstract of the work entitled “Characterizing networks with multiple arc-disjoint branching flows”, which is currently under review.
2. The full version of the work entitled “Coloring problems on bipartite graphs of small diameter” is available in the *Electronic Journal of Combinatorics (**E-JC**), volume 28 (2), 2021*, and at CoRR abs/2004.11173.
3. An extended abstract of the work entitled “Edge-disjoint branchings in temporal graphs” is available in *Proceedings of the 31st International Workshop on Combinatorial Algorithms (**IWOCA**), volume 12126 of LNCS, pages 112-125, 2020.* The full version is currently under review, and a preprint is available at CoRR abs/2002.12694.

Characterizing networks with multiple arc-disjoint branching flows*

Cláudio Carvalho[1] Jonas Costa[1] Cláudia Linhares Sales[1] Raul Lopes[1]
A. Karolinna Maia[1] Nicolas Nisse[2]

[1]Departamento de Computação, Universidade Federal do Ceará, Brasil
[2]Université Côte d'Azur, Inria, CNRS, I3S, Sophia-Antipolis, France

Abstract

An s-branching flow f in a network $\mathcal{N} = (D, u)$, such that u is the capacity function, is a flow that reaches every vertex in $V(D) \setminus \{s\}$ from s while loosing exactly one unit of flow in each vertex other than s. It is known that the hardness of the problem of finding k arc-disjoint s-branching flows in a network $\mathcal{N}$ is linked to the capacity u of the arcs in $\mathcal{N}$: for fixed c, the problem is solvable in polynomial time if every arc has capacity $n - c$ and, unless the Exponential Time Hypothesis (ETH) fails, there is no polynomial time algorithm for it for most other choices of the capacity function when every arc has the same capacity. The hardness of a few cases remains open. We further investigate a property depending on k and the capacity function that aims to characterize networks admitting k arc-disjoint s-branching flows. Such property is a generalization of similar result when all arcs have capacity $n - 1$, based on Edmonds' branching theorem. We show that, although in general this property doesn't offer a complete characterization, it characterizes particular cases of networks containing k arc-disjoint s-branching flows. For such positive cases, we also remark that there are polynomial-time algorithms to find the arc-disjoint flows, if they exist.

*This work is partly funded by FUNCAP Pronem 4543945/2016, CNPq project 304831/2017-4, STIC-AmSud project GALOP and the french Agence Nationale de la Recherche under contract Digraphs ANR-19-CE48-0013-01.
Emails: claudio@lia.ufc.br, jonascosta@lia.ufc.br, linhares@lia.ufc.br, raul@alu.ufc.br, karolmaia@ufc.br, nicolas.nisse@inria.fr

Coloring Problems on Bipartite Graphs of Small Diameter[*]

Victor A. Campos[1] Guilherme C. M. Gomes[2] Allen Ibiapina[3] Raul Lopes[1] Ignasi Sau[4] Ana Silva[3]

[1]Departamento de Computação, Universidade Federal do Ceará, Fortaleza, Brazil
[2]Departamento de Computação, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil
[3]Departamento de Matemática, Universidade Federal do Ceará, Fortaleza, Brazil
[4]LIRMM, Université de Montpellier, CNRS, Montpellier, France

Abstract

We investigate a number of coloring problems restricted to bipartite graphs with bounded diameter. First, we investigate the k-List Coloring, List k-Coloring, and k-Precoloring Extension problems on bipartite graphs with diameter at most d, proving NP-completeness in most cases, and leaving open only the List 3-Coloring and 3-Precoloring Extension problems when $d = 3$.

Some of these results are obtained through a proof that the Surjective C_6-Homomorphism problem is NP-complete on bipartite graphs with diameter at most four. Although the latter result has been already proved [Vikas, 2017], we present ours as an alternative simpler one. As a byproduct, we also get that 3-Biclique Partition is NP-complete. An attempt to prove this result was presented in [Fleischner, Mujuni, Paulusma, and Szeider, 2009], but there was a flaw in their proof, which we identify and discuss here.

Finally, we prove that the 3-Fall Coloring problem is NP-complete on bipartite graphs with diameter at most four, and prove that NP-completeness for diameter three would also imply NP-completeness of 3-Precoloring Extension on diameter three, thus closing the previously mentioned open cases. This would also answer a question posed in [Kratochvíl, Tuza, and Voigt, 2002].

[*]This work is partly supported by FUNCAP/CNPq/Brazil Project PRONEM PNE-0112-00061.01.00/16, CAPES-PRINT Institutional Internationalization Program, process 88887.468331/2019-00, French projects DEMOGRAPH (ANR-16-CE40-0028), ESIGMA (ANR-17-CE23-0010), ELIT (ANR-20-CE48-0008), and UTMA (ANR-20-CE92-0027), FUNCAP/CNPq/Brazil, Project PRONEM PNE-0112-00061.01.00/16, CNPq Universal 401519/2016-3 and 437841/2018-9, CNPq Produtividade 304576/2017-4.
Emails: victoitor@ufc.br, gcm.gomes@dcc.ufmg.br, allen.ibiapina@alu.ufc.br, raul@alu.ufc.br, ignasi.sau@lirmm.fr, anasilva@mat.ufc.br.

Edge-Disjoint Branchings in Temporal Digraphs[*]

Victor Campos[1] Raul Lopes[1] Andrea Marino[2] Ana Silva[3]

[1]Departamento de Computação, Universidade Federal do Ceará, Brazil
[2]Dipartimento di Sistemi, Informatica, Applicazioni, Università degli Studi di Firenze, Firenze, Italy
[3]Departamento de Matemática, Universidade Federal do Ceará, Fortaleza, CE, Brazil

Abstract

A temporal digraph $\mathcal{G}$ is a triple (G, γ, λ) where G is a digraph, γ is a function on $V(G)$ that tells us the time stamps when a vertex is active, and λ is a function on $E(G)$ that tells for each $uv \in E(G)$ when u and v are linked. Given a static digraph G, and a subset $R \subseteq V(G)$, a spanning branching with root R is a subdigraph of G that has exactly one path from R to each $v \in V(G)$. In this paper, we consider the temporal version of Edmonds' classical result about the problem of finding k edge-disjoint spanning branchings respectively rooted in given $R_1, \cdots, R_k$. We introduce and investigate different definitions of spanning branchings, and of edge-disjointness in the context of temporal digraphs. A branching $\mathcal{B}$ is vertex-spanning if the root is able to reach each vertex v of G at some time where v is active, while it is temporal-spanning if each v can be reached from the root at every time where v is active. On the other hand, two branchings $\mathcal{B}_1$ and $\mathcal{B}_2$ are edge-disjoint if they do not use the same edge of G, and are temporal-edge-disjoint if they can use the same edge of G but at different times. This lead us to four definitions of disjoint spanning branchings and we prove that, unlike the static case, only one of these can be computed in polynomial time, namely the temporal-edge-disjoint temporal-spanning branchings problem, while the other versions are NP-complete, even under very strict assumptions.

[*]Partially supported by FUNCAP/CNPq/Brazil, Project PRONEM PNE-0112-00061.01.00/16, CNPq Universal 401519/2016-3/ Produtividade 304576/2017-4, MIUR under PRIN Project n. 20174LF3T8 AHeAD (Efficient Algorithms for HArnessing Networked Data), and by the University of Florence under Project GRANTED (GRaph Algorithms for Networked TEmporal Data).
Emails: victoitor@ufc.br, raul@alu.ufc.br, andrea.marino@unifi.it, anasilva@mat.ufc.br.