Generalizing graph decompositions.

Benjamin Merlin Bumpus

Submitted in fulfilment of the requirements for the Degree of Doctor of Philosophy

School of Computing Science

College of Science and Engineering

University of Glasgow

Abstract

The Latin aphorism ‘divide et impera’ conveys a simple, but central idea in mathematics and computer science: ‘split your problem recursively into smaller parts, attack the parts, and conquer the whole’. There is a vast literature on how to do this on graphs. But often we need to compute on other structures (decorated graphs or perhaps algebraic objects such as groups) for which we do not have a wealth of decomposition methods. This thesis attacks this problem head on: we propose new decomposition methods in a variety of settings.

In the setting of directed graphs, we introduce a new tree-width analogue called *directed branch-width*. We show that parameterizing by directed branch-width allows us to obtain linear-time algorithms for problems such as directed Hamilton Path and Max-Cut which are intractable by any other known directed analogue of tree-width. In fact, the algorithmic success of our new measure is more far-reaching: by proving algorithmic meta-theorems parameterized by directed branch-width, we deduce linear-time algorithms for all problems expressable in a variant of monadic second-order logic.

Moving on from directed graphs, we then provide a meta-answer to the broader question of obtaining tree-width analogues for objects other than simple graphs. We do so introducing the theory of *spined categories* and *triangualtion functors* which constitutes a vast category-theoretic abstraction of a definition of tree-width due to Halin. Our theory acts as a black box for the definition and discovery of tree-width-like parameters in new settings: given a spined category as input, it yields an appropriate tree-width analogue as output.

Finally we study temporal graphs: these are graphs whose edges appear and disappear over time. Many problems on temporal graphs are intractable even when their topology is severely restricted (such as being a tree or even a star); thus, to be able to conquer, we need decompositions that take temporal information into account. We take these considerations to heart and define a purely temporal width measure called *interval-membership-width* which allows us to employ dynamic programming (i.e. divide and conquer) techniques on temporal graphs whose *times* are sufficiently well-structured, regardless of the underlying topology.

Table of Contents

List of Figures

Acknowledgements

At the dinner table, I was asked: ‘an aknowledgement section? Is that where you acknowledge that, despite all your reading, you still don’t know anything?’ I laughed, but I guess there’s some truth there.

These last three years have been - in many ways - transformative and I want to extend my gratitude to all those who have supported me along the way, to all those who have taught me and to all those who opened my eyes to that which I have yet to learn.

First of all, I want to thank my supervisor Kitty Meeks. Thank you for taking me on as a student, guiding me through the PhD with kindness and generosity and for teaching me how to explore and enquire with confidence. You made every step of the way as interesting and stimulating as it could be, so thank you.

I want to thank Ornela Dardha and Adam Kleczkowski for being my mentors and friends. I want to thank my second supervisor David Manlove for his thought-provoking questions and suggestions during my annual progression vivas. Thanks also to Jessica Enright, Donal Smith, Jòzsef Farkas, Marwan Fayed and everyone else who taught me mathematics. I want to thank my colleagues and office mates Sofiat, Frances, Craig, William, Michael and Ivaylo.

I want to thank my friends and colleagues Tom Wallis and Zoltan Kocsis. Thank you Tom for traveling around the world (often to other galaxies from the comfort of a coffee shop) with me: our countless adventures have shaped me in so many ways. Thank you Zoltan for always being there for me and for always being excited to learn and discover with me. I can’t keep track of the number of times we got caught in the rain while reading mathematics together in one of Stirling’s parks.

In my personal life, I want to thank my Mom, Dad, Chris, Francesca, Calvin and the rest of my family for their love and support and for teaching me to be true to myself: “lascia stare la porta, entra dalla finestra!” Finally, thank you Papoula. Your love, strength and support made the sun shine even when life was a storm. We might no-longer be Glasgow’s favourite pirates, but we still sail every sea together.

Author's declaration

I declare that, except where explicit reference is made to the contribution of others, that this dissertation is the result of my own work and has not been submitted for any other degree at the University of Glasgow or any other institution.

Printed Name: Benjamin Merlin Bumpus

Signature: 

To Papoula.

1 | Introduction

Graphs describe data and relationships therein. As such they are a fundamental mathematical tool that crops up often in mathematics, computer science and many other application areas ranging from sociology to chemistry to epidemiology and beyond. For example, we might use graphs to represent the atoms and their bonds in a chemical molecule or to describe people and their social contacts for an epidemiological study.

To extract information from graphs, we often need to probe them algorithmically. Unfortunately, since the vast majority of computational problems on graphs are NP-hard, we cannot expect to find algorithms that solve these problems quickly (i.e. in polynomial-time) on any input graph. In practical terms, this amounts to the fact that – more often than not – the best provably-correct algorithms that we know of run in exponential time with respect to the size of the input. Still more concretely, suppose for example that we have an algorithm (presumably one that answers some question which is very important to us) whose running time depends exponentially – say proportionally to 2^n – on the input size n. Suppose we can process n data points in a minute, if, after a few more days of data-collection, we collect another 20 data points, our algorithm now takes *over a year* to run. This apparent barrier of intractability is clearly a problem.

One way of coping with algorithmic intractability is by dropping the requirement of having running-time guarantees on any input graph: rather than considering algorithms on the class of *all* graphs, we restrict our attention to *specific subclasses*. For example, it is known that many problems that are intractable (NP-hard) in general are efficiently solvable on trees [48]. Since they do not have cycles, trees have an obvious recursive structure that lends itself nicely to the design of fast divide and conquer algorithms. In general, it is one of the great achievements of *parameterized complexity* to show that it is very often the case that 'recursive structure' (in a much more general sense than just trees) is algorithmically exploitable on graphs [26, 29, 43, 52]. The basic idea is that, on appropriately defined classes of graphs that display this recursive structure, one can design

efficient (often linear time) algorithms which proceed by dynamic programming (i.e. divide and conquer) even for problems that are NP-hard in the general case. In practical terms, this amounts to:

(1) splitting our graph into small parts which interact in simple ways,

(2) solving the problem on the parts, and

(3) deducing the solution recursively on the whole.

A huge body of work has been devoted to the study of the above steps (1), (2) and (3) when we are faced with *graph-based* computational problems [26, 29, 43, 52, 93]. In contrast, here we look *beyond graphs*. In particular, this thesis focuses on the first step: how should we split (in algorithmically-exploitable ways) different kinds of inputs into smaller parts?

Why should we look beyond graphs? As we mentioned, graphs are convenient ways of modeling real world systems. However, in applications one often needs to encode more information than just a binary relation. It might be natural to require edges to have *directions* [8, 9] for example when modeling of one-way roads in transportation networks or food webs where a directed edge represents the predator-prey relationship. Alternatively, we might need to represent the evolution of a graph over time [22, 79] (with edges appearing and disappearing). This might occur for example when representing a social network [22] (friendships may change with time) or a network in which edges represent signals between satellites which come-and-go depending on how close two satellites are to each-other in their orbits [22]. In fact it might even be that the best mathematical formalism for a particular application has little do with graphs: for example it might be an algebraic object such as a group or a vector space. Thus there is a strong need to *develop algorithmic tools – not just for graphs – but for other settings as well*.

This thesis should thus be understood as part of the larger scientific ambition of extending our current algorithmic techniques to new settings. Within this context, the present contribution is to take *tree decompositions* – one of the most well-understood [30, 74, 85–87, 89] and algorithmically powerful [26, 29, 42, 43, 52] methods of recursive decomposition on graphs – and propose similar notions (often generalizations) in a variety of new settings such as *digraphs*, *temporal graphs* and *abstract categories*.

Chapter Outline We shall defer the more detailed overview of the technical contributions of this thesis to Section 1.3 so that we can first briefly recall some standard graph- and complexity-theoretic notions in Section 1.1 as well as recalling the necessary background on tree-width in Section 1.2. We note that, since the techniques

and application domains of each of the main chapters differ greatly, we will introduce all of the more specialist terminology and background (e.g. relating to digraphs, category theory and temporal graphs) only once it is needed in the relevant chapters.

1.1 Preliminaries

We denote the *powerset* of any set S by 2^S. For any natural number n, we let $[n]$ denote the set $\{1, \dots, n\} \subseteq \mathbb{N}$ and we write $\binom{S}{n}$ to denote the set of all n-element subsets of any set S. We denote the *disjoint union* of two sets A and B by $A \uplus B$ and their *difference* as either $A \setminus B$ or $A - B$.

Graphs A *directed graph* (also called a *digraph*) is a set V called the *vertex-set* equipped with a binary *edge-relation* E; that is a pair (V, E) with $E \subseteq V^2$. Directed graphs in which the edge-relation is symmetric are called *graphs*. We say that two vertices are *adjacent* if they are joined by an edge while we say that two edges e and f are *incident* (written $e \cap f \neq \emptyset$) if they share an endpoint. For two vertices x and y we will simply write $\overrightarrow{xy}$ rather than (x, y) to denote a directed edge with *tail* x and *head* y; furthermore, slightly abusing notation, we shall consider undirected edges – which we denote as xy – as un-ordered pairs $\{x, y\}$ rather than as the relation $\{(x, y), (y, x)\}$. In a digraph, vertices which have outgoing edges, but no incoming edges are called *source* vertices and vertices that have incoming edges, but no outgoing edges are called *sink* vertices. A *multi-graph* (resp. *multi-digraph*) is a graph (resp. digraph) in which we allow multiple occurrences of any edge: we call two edges that have the same endpoints *parallel*. A *loop-edge* at a vertex x is an edge from the vertex x to itself. Graphs that do not contain any loop edges are called *simple graphs*. A *reflexive graph* (resp. *reflexive digraph*) is a graph (resp. digraph) in which there is a loop-edge at *every* vertex. A *hypergraph* is a pair (V, E) of vertices and *hyperedges* where every hyperedge $E \subseteq V$ is a vertex-subset. For $r \in \mathbb{N}$, we say that a hypergraph H is r-uniform if every one of its hyperedges has cardinality exactly r. Note that simple graphs are just 2-uniform hypergraphs. We write $V(H)$ and $E(H)$ to denote the set of vertices and edges of a hypergraph H and we extend this notation in an analogous way to graphs (be they simple or not) and digraphs.

The class $\mathcal{G}$ consists of all finite simple graphs (i.e. without loops or parallel edges). We write K_n to denote $([n], \{xy : x, y \in [n]\})$ and we call it the *n-vertex complete graph* (also referred to as '*n-vertex clique*'); K_0 is the empty graph $(\emptyset, \emptyset)$ and K_1 is the graph with a single vertex and no edges. Analogously, we refer to the hypergraph $([n], 2^{[n]} \setminus \{\emptyset\})$ as the *complete hypergraph*.

The *complement* of a graph G is the graph $\overline{G} := \left(V(G), \binom{V(G)}{2} \setminus E(G)\right)$ obtained by removing all edges in G and adding all edges that are not in G. We call $\overline{K_n}$ the *n-vertex discrete graph*. For graphs G and H, we denote by $G \uplus H$ and $G \cap H$ respectively their *disjoint union* $(V(G) \uplus V(H), E(G) \uplus E(H))$ and *intersection* $(V(G) \cap V(H), E(G) \cap E(H))$. We call a vertex v of a graph G an *apex* if it is adjacent to every other vertex in G. We denote by $G \star v$ the operation of adjoining a new apex vertex v to G.

An *n-edge walk* in a graph G is a finite alternating sequence $W := (x_1 e_1, \dots, x_n e_n x_{n+1})$ of vertices and edges of H such that $\{x_i, x_{i+1}\} \subseteq e_i$ for each $i \in [n]$. We call W a *circuit* if x_1 and x_{n+1} coincide. If no confusion arises, we will write walks and circuits only as sequences of edges (i.e. we might write $e_1, \dots, e_n$ to denote the walk W above). A walk (resp. circuit) with no repeated vertices is called a *path* (resp. a *cycle*). A graph that contains no circuits is called a *tree*. We say that a graph (resp. directed graph) is connected (resp. strongly connected) if there is a path (resp. directed path) connecting every pair of its vertices. A *vertex separator* (or simply a *separator*) in a connected graph G is a vertex-subset $S \subseteq G$ which, when removed from G yields a disconnected graph (i.e. S is a separator if $G - S$ is disconnected). The vertex-connectivity number of a graph G (denoted $\kappa(G)$) is the cardinality of the smallest separator in G (note that for a complete graph K_n, we follow the convention that $\kappa(K_n) = n$).

Definition 1.1.1. A *graph homomorphism* from a graph G to a graph H is a mapping $h : V(G) \to V(H)$ such that $h(x)h(y) \in E(H)$ whenever $xy \in E(G)$.

Throughout, given graphs G and H, the notation $G \hookrightarrow H$ will denote an injective graph homomorphism from G to H while the notation $G \twoheadrightarrow H$ will denote a surjective graph homomorphism from G to H. We say that two graphs G and H are *isomorphic* (denoted $G \cong H$) if there is a bijective graph homomorphism between them. We say that G is a *subgraph* of H (denoted $G \subseteq H$) if there is an injective graph homomorphism $\phi : G \hookrightarrow H$.

Given a graph G and a partition θ of its vertex set, we write $G/_\theta$ to denote the *quotient* of G with respect to θ; this is defined as the (not necessarily simple) graph obtained from G by identifying all vertices in each part of θ to a single vertex. Notice that θ defines a surjective graph homomorphism from G to $G/_\theta$; slightly abusing notation we will write this homomrphism as $\theta : G \twoheadrightarrow G/_\theta$.

The *clique-number* $\omega(G)$ of a graph G is the maximum-possible n such that there is an injective graph homorphism of the form $K_n \hookrightarrow G$. Alternatively $\omega(G)$ can be defined as the largest-possible clique that can be found in G as a subgraph.

The *chromatic number* $\chi(G)$ of a graph G is the minimum-possible $n \in \mathbb{N}$ such that there is a surjective

graph-homomorphism $G \to K_n$. Alternatively, $\chi(G)$ can be defined as the minimum number of colors needed in order to color every vertex of a graph in such a way that no two adjacent vertices are colored with the same color. To see why these two definitions coincide, notice that, since we are assuming that K_n has no loop edges, no two adjacent vertices in G can be mapped to the same color (i.e. vertex in K_n) by a homomorphism.

Given two graphs G and H, we say that G is a *minor* of H if G can be obtained from H via sequence of vertex deletions, edge deletions and edge contractions (an *edge contraction* of an edge e in some graph G is the operation of modifying G by identifying the two endpoints of e and deleting the resulting loop edge). The *Hadwiger number* of a graph G is the maximum n such that G has a K_n-minor.

The last graph-theoretic notion that we will need for now is that of an H*-sum* of two graphs. The informal idea is that the H-sum of two graphs G_1 and G_2 is computed by finding isomorphic copies of a fixed subgraph H in both of them and then constructing an H-sum of G_1 and G_2 by identifying the vertices of H in G_1 to the vertices of H in G_2 and removing any parallel edges. The formal definition is as follows (alternatively, see Figure 1.1).

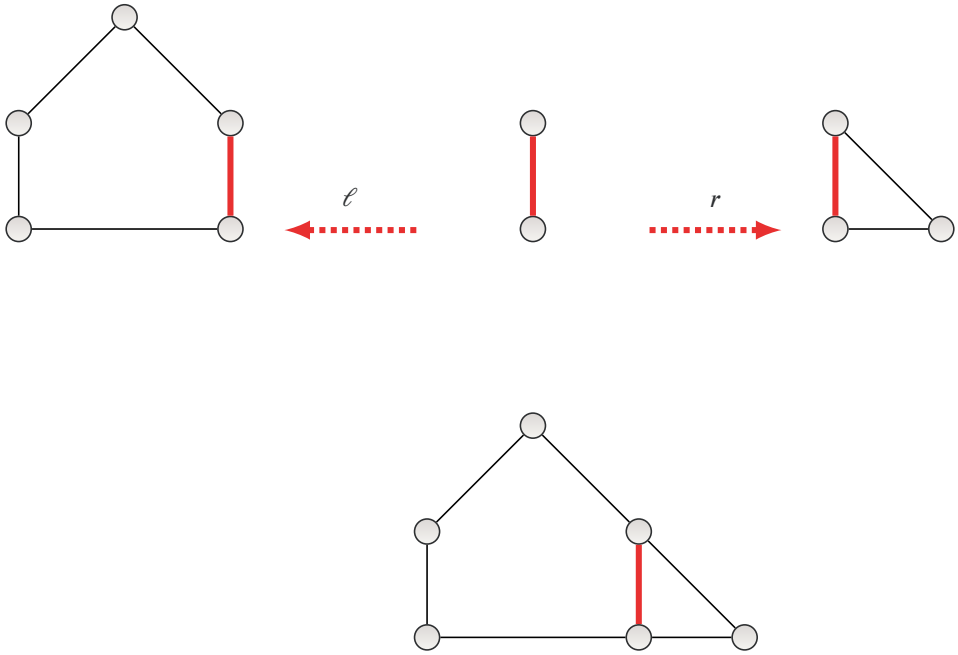


Figure 1.1: A graph $C_5 \#_{K_2} K_3$ (center bottom) constructed according to the injections $\ell : K_2 \to C_5$ (top left) and $r : K_2 \to K_3$ (top right).

Definition 1.1.2 ([55], see also [30])**.** Given injective homorphisms $h_1 : H \to G_1$ and $h_2 : H \to G_2$, the H*-sum* of G_1 and G_2 along (h_1, h_2) is the graph $G_1 \#_H G_2$[1] defined as

$$G_1 \#_H G_2 := \left(V(G_1) \uplus V(G_2) /_{h_1^{-1} = h_2^{-1}}, E(G_1) \uplus E(G_2) /_{\sim} \right)$$

where any two edges $h_1(w)h_1(x)$ and $h_2(y)h_2(z)$ in the ranges of h_1 and h_2 respectively are related under $\sim$ if the edges wx and yz are equal in H.

For any graph-theoretic notion not defined here, we refer the reader to Diestel's textbook [30].

Complexity A language over an alphabet Σ is a subset $\Sigma' \subseteq \Sigma^*$ of the set Σ^* of all finite words over Σ (i.e. Σ^* is the free monoid generated by Σ under concatenation). A *classical decision problem* $\mathfrak{R}$ is a language over some fixed alphabet Σ. We call a string I in Σ^* an *instance* and in particular we call it a *yes-instance* if $I \in \mathfrak{R}$ and a *no-instance* otherwise.

We say that a problem $\mathfrak{R}$ is in P if it is recognised in polynomial time by a deterministic Turing machine. Similarly, we say that $\mathfrak{R}$ is in NP if it is recognised in polynomial time by a non-deterministic Turing machine. Alternatively, the class NP consists of all problems admitting certificates that are checkable in polynomial time by a deterministic Turing machine. A problem $\mathfrak{R}$ is NP-complete if it is in NP and if it is NP-hard (recall that $\mathfrak{R}$ is NP-hard if every problem in NP admits a polynomial-time computable reduction to $\mathfrak{R}$). We refer the reader to Garey and Johnson's textbook [48] for a more in-depth treatment.

Given a problem $\mathfrak{R}$, an *advice to* $\mathfrak{R}$ is an $\mathbb{N}$-indexed sequence $(a_1, a_2, \dots)$ of strings over the alphabet of $\mathfrak{R}$. A Turing machine M (be it deterministic or not) for $\mathfrak{R}$ is said to *compute with advice* $(a_1, a_2, \dots)$ if, given any instance I of $\mathfrak{R}$, M receives as input both I and the $|I|$-th advice string $a_{|I|}$ (note that $a_{|I|}$ depends only on $|I|$). We say that a problem $\mathfrak{R}$ is in P/POLY if it is recognised in polynomial time by a deterministic Turing machine computing with any advice $(a_1, a_2, \dots)$ satisfying the requirement that there is a polynomial p such that $|a_n| \leq p(n)$ for all naturals n (we will make use of the complexity class P/POLY to state a complexity theoretic hypothesis in Chapter 2.)

Parameterizations Rather than measuring the running time of (graph) algorithms solely with respect to the input size, we might also keep track of the dependence on some 'structural parameter' of the input (graph). This approach belongs to the vast field of *Parameterized Complexity*. We will first convey the rough intuition and defer formal definitions. The idea is that, given some decision problem $\mathcal{Q}$ on a graph and a measurement $\mu : \mathcal{G} \to \mathbb{N}$ which assigns a positive integer value to each finite simple graph, we measure the running time of algorithms on any given graph G, not only with respect to the input size $|G|$, but also with respect to the *paramterization* $\mu(G)$. Of particular interest are algorithms that run in time $f(\mu(G)) \cdot |G|^c$ where f is any computable function and c is

[1] The reader might rightly object that this operation should be written as $G_1 \#_{h_1,h_2} G_2$ since different choices of injective homomorphisms do indeed yield a different H-sums. We choose to write $G_1 \#_H G_2$ in keeping with the notation of both Halin [55] and Diestel [30]. This notation will only be used in settings in which there is no ambiguity.

a constant. Notice that algorithms with running times of this kind yield polynomial-time algoritms for Q on any class of the form $\{G \in \mathcal{G} : \mu(G) \leq k\}$ for some *fixed* positive integer k. We state these ideas more formally as follows.

Definition 1.1.3. Take a fixed alphabet Σ and a mapping $\kappa : \Sigma^* \to \mathbb{N}$ called a *parameterization*. We define a *parameterized decision problem* to be a pair $(\mathfrak{R}, \kappa)$ where $\mathfrak{R} \subseteq \Sigma^*$ is a language over Σ. Given any string I in Σ^*, we call the pair $(I, \kappa(I))$ an *instance* of the parameterized problem $(\mathfrak{R}, \kappa)$.

A parameterized problem $(\mathfrak{R}, \kappa)$ is *slice-wise polynomial tractable* if there exists an algorithm A which, for any instance $(I, \kappa(I))$, decides whether I lies in $\mathfrak{R}$ in time at most $f(\kappa(I))|I|^{g(\kappa(I))}$, where f and g are computable functions. If g is a constant function, then we call A a *fixed-parameter algorithm* and we call any parameterized problem admitting such an algorithm *fixed parameter tractable*. The class of all slice-wise polynomial tractable problems is denoted XP and the class of all fixed parameter tractable problems is denoted FPT.

We say that a parameterized problem $(\mathfrak{R}, \kappa)$ is *para*-NP-*hard* if there exists some fixed constant d, such that it is NP-hard to decide any instance I of $\mathfrak{R}$ with $\kappa(I) = d$. Notice that, unless P = NP, showing para-NP-hardness is a way of ruling out the existence of any FPT algorithm for $(\mathfrak{R}, \kappa)$. To see this, notice that, if we had an algorithm A for the parameterized problem $(\mathfrak{R}, \kappa)$ running in time $f(\kappa(I))|I|^c$ (where f is a computable function and c a constant), then A would run in polynomial time (to be precise, it would run in time $f(d)|I|^c$) for the class of all instances I with $\kappa(I) = d$. However, this would then imply P = NP.

Another way of providing evidence that FPT-time algorithms for some parameterized problem $(\mathfrak{R}, \kappa)$ are unlikely to exist is by proving that this problem is 'hard' (we leave this undefined for now) with respect to one of the complexity classes in the so-called '**W**-*hierarchy*: this is a sequence of nested parameterized complexity classes $\mathbf{W}[1] \subseteq \mathbf{W}[2] \subseteq \mathbf{W}[3] \subseteq \cdots \subseteq W[P]$. Since in this thesis we will only concern ourselves with showing that a parameterized problem is likely not in FPT, we will not explain the distinction between the classes in the **W**-hierarchy (for which we refer the reader to the textbook by Cygan et al. [29]). Instead we will only give an explanation of what it means for a problem to be **W**[1]-hard. To do so, we first need a suitable notion of *parameterized reduction* (Definition 1.1.4) and a suitable assumption of intractability (Definition 1.1.5).

Definition 1.1.4. Let Σ and Σ' be alphabets over which two parameterized problems $(\mathfrak{R}, \kappa)$ and $(\mathfrak{R}', \kappa')$ are respectively defined. A *parameterized reduction* from $(\mathfrak{R}, \kappa)$ to $(\mathfrak{R}', \kappa')$ is a mapping $\rho : \Sigma \to \Sigma'$ satisfying the folloiwing three requirements:

- ρ is computable by an FPT-time algorithm with respect to the parameterization κ,
- for any $I \in \Sigma$, we have that $I \in \mathfrak{R}$ if and only if $\rho(I) \in \mathfrak{R}'$,
- there is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that, for all $I \in \Sigma^*$, we have $(\kappa' \circ \rho)(I) \leq (g \circ \kappa)(I)$.

Given the notion of a parameterized reduction, we can now describe (Definition 1.1.5) a complexity theoretic hypothesis which can be used to define **W**[**1**]-hard problems; this hypothesis states that there is no FPT algorithm for k-CLIQUE when parameterized by k.

k-**CLIQUE**

Input: a graph G and an integer k.

Question: does D contain a clique on at least k vertices as a subgraph?

Definition 1.1.5 (Engineer's Hypothesis)**.** k-CLIQUE parameterized by k admits no algorithm with worst-case running time of the form $f(k)|V(G)|^c$ (where f is a computable function and c a constant).

Given this hypothesis and the notion of a parameterized reduction, we can finally define **W**[1]-hard problems as those problems that are, in the parameterized sense, 'at least as hard' as k-CLIQUE parameterized by k.

Definition 1.1.6. A parameterized problem $(\mathfrak{R}, \kappa)$ is $W[1]$-*hard* if there is a parameterized reduction from k-CLIQUE parameterized by k to $(\mathfrak{R}, \kappa)$.

We refer the reader to either Cygan et. al.'s [29] of Flum and Grohe's textbooks [43] for other (more standard) definitions of the **W**-hierarchy (for example definitions in terms of circuits and their *wheft*).

1.2 Tree-width

Tree-width is a function $\mathbf{tw} : \mathcal{G} \to \mathbb{N}$ which – at an intuitive level – measures how similar the global connectivity of a given graph is to that of a tree. For example, forests with at least one edge have tree-width 1 and, for $n > 1$, n-vertex cliques have tree-width $n - 1$. Rather than defining it immediately, we shall take a step back and first gain some intuition about why one should expect to be able to solve problems efficiently on graphs that are 'tree-like' and then we shall see how this intuition relates to the definition of tree-width.

Exploiting recursive structure As we already mentioned, many NP-hard problems are tractable on classes of recursively decomposable objects. In fact this intuition was already noticed by Johnson in 1985 in his "16th NP-completeness column" [62]. There he compiled a list of graph classes for which many NP-complete problems are polynomial- or even linear-time solvable and observed that the members of many of these classes can be decomposed (or dually built-up from smaller parts) in a recursive way. Some examples from Johnson's list are: trees, chordal graphs, partial k-trees, series parallel graphs, split graphs and co-graphs [62].

Out of Johnson's list we will single-out *chordal graphs*. These are often defined as graphs in which all cycles of length at least 4 have a chord (i.e. an edge connecting two vertices of the cycle, but which is not used by the cycle itself). So far we have mentioned that chordal graphs have a recursive structure, but this structure is not apparent from the definition we just gave; thus we will present the following equivalent definition of chordal graphs which is due to Dirac [34].

Definition 1.2.1 (Dirac's Theorem for chordal graphs [34])**.** We define the class of all *chordal graphs* recusively as follows:

- every complete graph K_n is a chordal graph
- given two chordal graphs G_1 and G_2 and any $n \in \mathbb{N}$ such that G_1 and G_2 both contain a copy of a complete graph K_n as a subgraph, any clique-sum of the form $G_1 \#_{K_n} G_2$ is a chordal graph.

The recursive structure of chordal graphs can be exploited algorithmically to obtain polynomial- (or even linear-) time algorithms for many NP-hard problems (we refer the reader to Golumbic's textbook [51] for an in-depth treatment). As an example, we show (Example 1.2.2) how to determine the chromatic number of a chordal graph in polynomial time (in contrast, note that, for any $k \geq 3$, the general problem of determining whether a graph has chromatic number at most k is NP-complete).

Example 1.2.2 (coloring chordal graphs)**.** *Notice that, in a chordal graph H, any inclusion-wise maximal clique in H must be a vertex-separator (meaning that its removal disconnects H).*

Thus, since finding an inclusion-wise-maximal clique can be done in polynomial time, either H is itself a clique, or we can re-write (in polynomial time) H as a clique-sum of the form $H = H_1 \#_{K_i} H_2$ where i is the size of the maximal clique we found and H_1 and H_2 are two chordal graphs strictly smaller than H (i.e. $|V(H_i)| < |V(H)|$ for $i \in [2]$). This leads us to the following algorithm (we call it **findChrom***) which we describe recursively as follows.*

$$\textbf{findChrom}(H) := \begin{cases} |V(H)| & \text{if } H \text{ is a clique} \\ \max\{\textbf{findChrom}(H_1), \textbf{findChrom}(H_2)\} & \text{if } H = H_1 \#_{K_i} H_2. \end{cases}$$

Since $\chi(K_n) = n$, $\forall n \in \mathbb{N}$, the proof of correctness of the **findChrom** *algorithm follows by induction by noticing that, for any two graphs (not even necessarily chordal ones) G_1 and G_2 and any suitable n, we have $\chi(G_1 \#_{K_n} G_2) = \max\{\chi(G_1), \chi(G_2)\}$. To see why this is, properly color G_1, properly color G_2 and then permute the labels of the color classes on G_2 so that the colors of the vertices of K_n in G_1 match those of K_n in G_2. This must always be possible since, up to permutations of the labels of color classes, K_n has a unique proper coloring.*

Tree-width was introduced independently by many authors [11,55,85] and it has accumulated many equivalent, but often cryptomorphic definitions each exhibiting different structural correspondences and insights [11, 26, 53, 55, 85]. The choice of which definition to present often boils down to choosing whichever definition is most convenient for the task at hand. In practice, tree-width is most often defined in terms of the related concept of *tree decompositions* (Definition 1.2.3). These are ways of arranging the vertices and edges of a graph in a tree-like way. One might think of them as auxiliary data structures which can be used to exhibit a recursive pattern in the graph we are studying in an algorithmically useful way (see Figure 1.2). We will slightly postpone this definition of tree-width (Definitions 1.2.4) in order to first present an alternative definition in terms of (homomorphisms to) chordal graphs (Equation 1.1). We do this with the intent of providing the reader with a solid understanding of tree-width coupled with an intuitive justification of why tree-width should indeed be seen as a measure of how much the global connectivity of a graph differs from that of a tree.

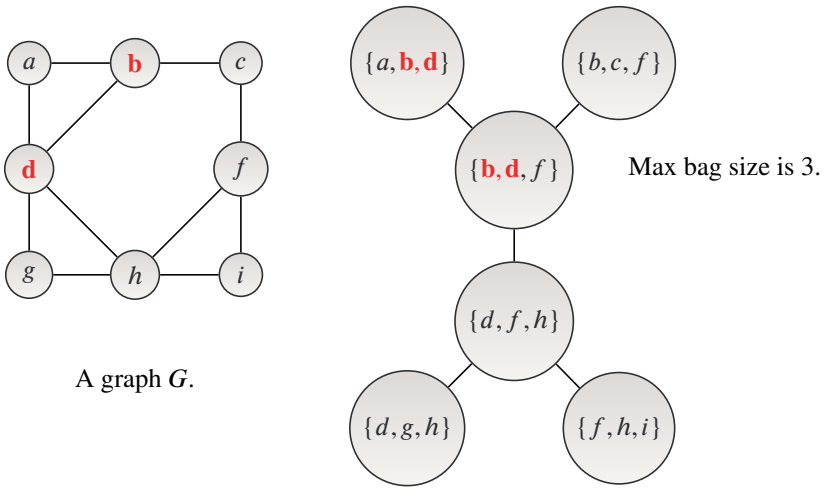
Imagine for a moment that, after hopping in a time machine, it was up to us to come-up with the first definition of tree-width. Having just studied chordal graphs, we notice that all forests (i.e. disjoint unions of trees) are chordal graphs since we can build them recursively by starting with the set $\{K_0, K_1, K_2\}$ and taking clique-sums along any element of $\{K_0, K_1, K_2\}$. It thus makes intuitive sense to think of the set of all chordal graphs which can be constructed using only elements of $\{K_0, K_1, \ldots, K_n\}$ to be considered more tree-like (i.e. they have a slimmer tree-like structure) than the corresponding set constructed using only elements of $\{K_0, K_1, \ldots, K_n\} \cup \{K_{n+1}\}$. More formally, this suggests that the clique-number ω might be a good measure of

how 'close' a chordal graph is to being a tree (in particular a chordal graph H is a forest if and only if $\omega(H) \leq 2$). Naturally, since we would like such a measure of 'tree-likeness' of a chordal graph to be 1 (rather than 2) for trees, one might define such a measure – let's call it **chordal-tree-likeness** – as the map $H \mapsto \omega(H) - 1$ associating each chordal graph to one less than its clique-number.

Now, armed with this notion of **chordal-tree-likeness**, we seek an analogous notion for any (not necessarily chordal) graph G. We will do so by simply reducing the general case to the chordal case: intuitively, our approach will be to find a 'most efficient' approximation of G via a chordal graph H and then determine how 'tree-like' H is (which, as we just saw, we can do by computing $\omega(H) - 1$). It turns out that formalizing these intuitions actually yields the first definition of *tree-width* that we encounter in this thesis; namely we have

$$\mathbf{tw}(G) := \min\{\omega(H) - 1 : G \text{ is a subgraph of a chordal graph } H\} \quad (\text{where } \omega \text{ is the clique-number}). \quad (1.1)$$

The definition we just saw (i.e. Equation 1.1) is due to Halin [55]. Tree-width is, however, most commonly defined in terms of the related concept of a *tree-decomposition* which is defined as follows (see also Figure 1.2).



A graph G.

A tree decomposition $(T, (V_t)_{t \in T})$ of G of width 2.

Figure 1.2: An example of a tree decomposition. The intersection $\{b, d\} = \{a, b, d\} \cap \{b, d, f\}$ (marked in **bold red**) of the two top-leftmost bags $\{a, b, d\}$ and $\{b, d, f\}$ of $(T, \mathcal{V})$ is a vertex-separator in G.

Definition 1.2.3. [11, 55, 88] The pair $(T, (B_t)_{t \in V(T)})$ is a *tree decomposition* of a graph (resp. hypergraph [1]) H if $(B_t)_{t \in V(T)}$ is a sequence of subsets of $V(H)$ (called *bags*) indexed by the nodes of the tree T such that:

(T1) for every edge (resp. hyper-edge) f of H, there is a node $t \in V(T)$ such that $f \subseteq B_t$,

(T2) for every $x \in V(H)$, the set $V_{(T,x)} := \{t \in V(T) : x \in B_t\}$ induces a *non-empty connected* subgraph in T.

Letting the *width* of a tree decomposition $(T, (V_t)_{t \in T})$ of the graph (resp. hypergraph) H be defined as one less than the maximum of the cardinalities of its bags, we have the following definition of tree-width.

Definition 1.2.4. The *tree-width* $\mathbf{tw}(H)$ of any graph or hypergraph H is the minimum possible width of any tree decomposition of H.

It is a standard exercise (for example see the textbooks by Diestel [30] or Cygan et. al. [29]) to show that the two definitions we just mentioned (i.e. Equation 1.1 and Definition 1.2.4) do indeed coincide. To see this, the key intuition is that, given some tree decomposition $(T, \mathcal{V})$ of G, if we add an edge (unless one is already present) between any two vertices in G that appear together in some bag in $\mathcal{V}$, then the resulting graph G' must be chordal (for full details see Diestel's textbook [30]).

Algorithms on classes of bounded-tree-width As we mentioned earlier, many computational problems that are NP-hard in general turn out to be tractable on trees and chordal graphs [48, 51, 62]. Indeed this pattern is continued more generally when we consider classes of bounded tree-width: there are many NP-hard problems (e.g. coloring, Hamiltonicity, finding Steiner trees etc. [29]) that are tractable on classes of graphs of bounded tree-width.

Intuitively, the bags of tree-decompositions play a role largely analogous to that of cliques in Example 1.2.2: given an instance G of some decision problem $\mathcal{Q}$ and a tree-decomposition $(T, \mathcal{V})$ of G, we solve our problem by brute force on the bags of $(T, \mathcal{V})$ and then we join the solutions together to obtain a solution for $\mathcal{Q}$ on G. Without going into detail (for which we instead refer the reader to any parameterized complexity textbook [26, 29, 43, 52]), we highlight that the crucial point is that algorithms of this kind often run in time $f(\mathbf{tw}(G))|G|^c$ for some computable functions f and constant c. In fact, very often we have $c = 1$ making these *linear-time* algorithms on classes of graphs of bounded tree-width [26, 29, 43, 52].

Perhaps more surprising is the sheer quantity[2] of algorithmic applications of tree-width [26, 29, 43, 52]. This abundance of tractability results on classes of bounded tree-width is partially explained by a deep connection between tree-decompositions, monadic second-order logic and tree-automata which is known as Courcelle's Theorem (Theorem 1.2.7). In order to state this theorem, we shall first recall some preliminary definitions from

[2] As a very rough estimate, a Glasgow University Library search (in May of 2021) for '`((tree-width) OR (treewidth) OR (tree decomposition))`' returns about 159'157 results

logic and finite model theory. We shall follow the notation found in Libkin's textbook [73] to which we refer the reader for any definitions in finite model theory not stated here; alternatively, for an introduction to model-theoretic methods in parameterized complexity theory we refer the reader to Flum and Grohe's textbook [43].

Logical preliminaries A *vocabulary* is a collection σ of *constant symbols* $c_1, c_2, \ldots$, *relation symbols* $E_1, E_2, \ldots$ and *function symbols* $f_1, f_2, \ldots$. Each symbol s in σ is associated with an element $\mathbf{ar}(s)$ of $\mathbb{N}$ called the *arity* of s and all constant symbols have arity zero. We call σ *relational* if it does not contain any function symbols.

Definition 1.2.5. For a vocabulary σ, a *σ-structure* (or *model*) is a four-tuple $\mathfrak{M} := (V, \{c_i^{\mathfrak{M}}\}, \{E_i^{\mathfrak{M}}\}, \{f_i^{\mathfrak{M}}\})$ consisting of a *universe* V together with an *interpretation* (i.e. a pairing) of

- each constant symbol c_i from σ as an element $c_i^{\mathfrak{M}} \in V$;
- each k-ary relation symbol E_i from σ as a k-ary relation $E_i^{\mathfrak{M}} \subseteq V^k$ on V;
- each k-ary function symbol f_i from σ as a k-ary function $f_i^{\mathfrak{M}} : V^k \to V$.

We call a σ-structure *finite* if its universe is finite and we will call it a *relational structure* if σ is relational. At an intuitive level, relational structures are just hypergraphs with labels on edges and labels on some distinguished vertices (corresponding to the constants). For example, taking σ to be a relational vocabulary with no constant symbols and only one r-ary relation symbol E, it is easy to see that r-uniform hypergraphs (where the universe is their vertex-set) are σ-structures. If we take E to be binary, then we have an encoding of graphs as relational structures. Alternatively, we could encode any graph G as a relational structure with universe $V(G) \cup E(G)$ under a binary *incidence* relation I^G (i.e. we take $\sigma = \{I\}$) which records which edges are incident with which vertices.

Now we define first-order (**FO**) formulae for relational vocabularies as follows.

Definition 1.2.6. Given a relational vocabulary σ and a countably-infinite set of variables $x, y, \ldots$, we call any variable and every constant symbol a *term*. The set $\mathbf{FO}[\sigma]$ of *first-order formulae over* σ is defined inductively as follows:

- if t_1 and t_2 are terms, then $t_1 = t_2$ is an *atomic* **FO**-formula over σ,
- if $t_1, \ldots, t_k$ are terms and E is a k-ary relation in σ, then $E(t_1, \ldots, t_k)$ is an *atomic* **FO**-formula over σ,

- if ϕ and ψ are **FO**-formulae over σ, then so are $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \to \psi$ and $\neg\phi$,
- if x is a variable and ϕ is a **FO**-formula over σ, then so are $\exists x\phi$ and $\forall x\phi$ (in this case we say that x is *bound* by a quantifier).

We call a variable x in a formula ϕ *free* if it is not bound by an existential or a universal quantifier. We say that a σ-structure $\mathfrak{M}$ *models* (or *is a model of*) a $\mathbf{FO}[\sigma]$-formula ϕ with free variables $x_1, \ldots, x_k$ if there are elements $a_1, \ldots, a_k$ in the universe of $\mathfrak{M}$ such that evaluating ϕ in $\mathfrak{M}$ under the substitution $x_i \mapsto a_i$ yields a true statement in **FO** logic. We use the shorthand $\phi(x_1, \ldots, x_k)$ to denote a formula ϕ with free variables $x_1, \ldots, x_k$ and, for $a_1, \ldots, a_k \in V(\mathfrak{M})$ in the universe of $\mathfrak{M}$, we write $\mathfrak{M} \vDash \phi(a_1, \ldots, a_k)$ if the substitution $x_i \mapsto a_i$ witnesses that $\mathfrak{M}$ models ϕ. (The notation $\phi(x_1, \ldots, x_k)$ highlights the fact that a **FO**-formula ϕ with k free variables can be seen as a function $\phi : V(\mathfrak{M})^k \to \{\top, \bot\}$ taking k-tuples of elements of the universe of $\mathfrak{M}$ to either true or false.)

As we have just seen, when we evaluate $\mathbf{FO}[\sigma]$-formulae (for any σ) on any σ-structure $\mathfrak{M}$, quantification (and hence also the choice for free variables) ranges over the elements of the universe $V(\mathfrak{M})$. However, when dealing with graphs, one is often interested in defining logical formulae which can reason about *subsets* of the universe $V(\mathfrak{M})$ as well. This sort of reasoning is captured by *second-order* logic (where we are allowed to quantify over both the universe and the relations).

Monadic second-order logic (denoted **MSO**) is the fragment of second-order logic which allows second-order quantification only over unary realtions (i.e. subsets of the universe). For graphs, monadic second-order quantification can mean different things depending on whether we encode graphs as:

1. a universe $V(G)$ of vertices and a binary edge relation E^G, or
2. a universe $V(G) \cup E(G)$ of vertices and edges and a binary incidence relation I^G.

In the first case, **MSO** quantification allows us only to quantify over vertex-subsets; in the second case, we are allowed to quantify over *both* vertex- and edge-subsets (since the universe is $V(G) \cup E(G)$ rather than just $V(G)$). To distinguish which encoding is assumed, we write $\mathbf{MSO}_1$-logic when we allow second-order quantification only over vertex-sets while we write $\mathbf{MSO}_2$-logic when we allow second-order quantification both over vertex-sets and edge-sets.

We say that a decision problem $\mathcal{Q}$ is *expressible* in **FO** (resp. $\mathbf{MSO}_1$ or $\mathbf{MSO}_2$) there exists a formula $\phi_{\mathcal{Q}} \in \mathbf{FO}$ (resp. $\mathbf{MSO}_1$ or $\mathbf{MSO}_2$) such that, for all graphs G, $G \vDash \phi_{\mathcal{Q}}$ if and only if G is a yes-instance of $\mathcal{Q}$.

Note that many (NP-hard) decision problems on graphs (such as, for some fixed k, whether the chromatic number is at most k or whether a given graph admits a Hamiltonian path etc. [29, 43]) are expressible in $\mathbf{MSO}_2$-logic.

The relationship between tree-width and MSO-logic for which we have been setting the scene is described by Courcelle's Theorem (Theorem 1.2.7). Roughly, this theorem states that, if we are given *any* decision problem $\mathcal{Q}$ that is $\mathbf{MSO}_2$-expressible via a formula $\phi_{\mathcal{Q}}$, then there is an FPT-algorithm which decides whether $G \models \phi_{\mathcal{Q}}$ (i.e. whether G is a yes-instance for $\mathcal{Q}$) when parameterized by *both* $\mathbf{tw}(G)$ and the length of ϕ.

Theorem 1.2.7 (Courcelle's Theorem, [25])**.** *There is an algorithm which, given a graph G of tree-width k and an $\mathbf{MSO}_2$-formula ϕ, decides whether $G \models \phi$ in linear time parameterized by both k and $|\phi|$.*

1.3 Overview of contributions

This thesis consists of three main chapters each proposing decomposition methods and their associated width-measures similar to (and often generalizing) tree decompositions in different settings. We begin with **directed graphs** in Chapter 2, then we consider **categories** in Chapter 3 and finally **temporal graphs** in Chapter 4.

Directed graphs Since the 1990s researchers have been searching for a notion analogous to tree-width which describes algorithmically useful recursive structure in directed graphs [68]. Over the past 30 years, a plethora of subtly different directed analogues of tree-width have been proposed, but, although they were all useful in certain application domains (and especially in structural settings [49, 63, 68]) they all suffer from significant algorithmic shortcomings [47, 68]. For example the directed variants of the Hamilton Path and Max-Cut problems are hard on any such class of 'bounded width' [45, 69]. At first glance this is not surprising, since it is known that the directed variants of most problems tend to be harder than their undirected counterparts [48, 68]. However, there are cases in which adding directions to the edges can actually make problem instances computationally simpler. For example, although it is NP-hard to determine if a graph admits a Hamiltonian path, if the input is an *acyclic directed* graph, then the problem can be solved in polynomial time. At the same time, though, there are problems (such as Max-Cut) that remain NP-hard on directed acyclic graphs. These considerations motivate the search for a truly directed analogue of tree-width that can distinguish cases in which edge-orientations can make problems tractable. The known directed analogues of tree-width do not capture these subtle points since they are all either bounded on the class of all directed acyclic graphs (e.g. directed tree-width, DAG-width etc. [68]) or they are

lower-bounded by their undirected counterparts (e.g. directed clique-width and directed rank-width [68]). At a rough, intuitive level, these observations can be read as: 'the known width-measures cannot distinguish cases in which adding directions to the edges can simplify problem instances'.

In contrast we introduce a new tree-width analogue in Chapter 2 called *directed branch-width* which *does* indeed distinguish many cases in which adding directions to the edges simplifies problem instances. In fact we find that directed branch-width enjoys many of the algorithmic properties that one might hope for in a directed analogue of tree-width: in particular we show that many NP-hard problems (including the directed versions of Hamilton path and Max-Cut) are tractable on digraph classes of bounded directed branch-width. On the structural side, we prove a characterization of classes of bounded directed branch-width in terms of the bi-cut-rank-width (roughly this is a measure of decomposability via low-rank cuts) of their corresponding class of directed line-graphs. Furthermore, we find that, although classes of bounded directed branch-width need not have bounded underlying tree-width, directed branch-width differs from underlying tree-width only on source and sink vertices. These results allow us to establish a much more general algorithmic meta-theorem parameterized by directed branch-width. This theorem proves the existence of linear-time algorithms for all problems expressable in a restricted variant of monadic second-order logic. (Chapter 2 is based on joint work with Meeks and Pettersson [19].)

An abstract analogue of tree-width Having proposed algorithmically useful decomposition methods for directed graphs, we move to higher abstractions in Chapter 3 by adopting a category-theoretic perspective which proposes a meta-answer to the question of obtaining tree-width analogues for objects other than simple graphs. Our starting point is Halin's characterization of tree-width as the maximal function sharing certain properties with the vertex-connectivity number, the Hadwiger number and the chromatic number [55]. Out of the many cryptomorphic definitions of tree-width [11, 26, 53, 55, 85], Halin's definition has a much more algebraic flavour which is ripe for abstraction. In fact, by thinking compositionally, we obtain a vast generalization of Halin's result to arbitrary categories by introducing the theory of *spined categories* and *triangulation functors* in Chapter 3. Our algebraic formulation allows for great generality since it can be thought as a black-box which yields the definition of an appropriate tree-width analogue whenever it is fed as input a class of objects equipped with a suitable notion of containment. As such, the theory of spined categories provides a blueprint for defining new tree-width analogues in diverse settings by simply collecting the objects of interest into an appropriate spined category. (Chapter 3 is based on joint work with Kocsis [17].)

Temporal graphs If the search for good tree-width analogues for directed graphs was challenging, then it is natural to expect this question to be even harder when we move from the setting of directed graphs to that of *dynamic*, *evolving* or *temporal graphs*. These are graphs whose edges appear and disappear over time and which are used to model many real-world phenomena such as disease spread, trade, social contacts etc. [22, 79]. In this setting traditional notions of tree-width fail to be of much algorithmic use since many computational problems remain hard even on temporal trees [3, 77]. It is thus apparent that, to obtain a useful notion of temporal structure on temporal graphs, we need notions of *temporal* connectivity and *temporal decomposition* that do not depend solely on the underlying graph-theoretic structure.

In Chapter 4 we determine the computational complexity of some natural edge-exploration problems on temporal graphs. In particular we prove hardness results yielding further examples of natural problems that are hard when the underlying graph has a very restricted structure (tree-width at most 2 or vertex-cover number at most 2). This motivates the development of a parameter unrelated to the treewidth of the underlying graph. To that end, we introduce a new *purely temporal* width-measure called *interval-membership-width*. This measure depends solely on the temporal structure of the input temporal graph and not on the underlying graph-theoretic structure and, via a connection to interval graphs, we find it to be reminiscent of the well-known notion of *path-width* (where path-width is defined by taking the definition of tree-width and imposing upon it the additional requirement that the indexing trees of all decompositions must be paths).

Interval-membership-width gives us just enough structural insight to be able to formulate dynamic-programming algorithms for two problems on temporal graphs (one of which is determining whether a temporal graph is temporally Eulerian) that remain hard even on temporal graphs whose underlying static graph is a tree or a cactus graph. Our algorithms run in linear-time on classes of bounded interval-membership-width and we believe that in the future similar results are likely to follow for other connectivity-based temporal problems as well (e.g. problems related to spreading processes, reachability or exploration). (Chapter 4 is based on joint work with Meeks [18].)

2 | Directed branch-width: a directed analogue of tree-width

2.1 Introduction

It is often the case that problems on directed graphs are tractable on classes of inputs whose underlying class of undirected graphs (i.e. ignore edge directions) have bounded tree-width [68]. However, there are situations in which certain orientations of edges can make some problems tractable even when they are intractable on the underlying undirected graph classes. For example, the Hamilton Path problem is NP-complete on undirected graphs but it is polynomial-time solvable on any acyclic orientation of any graph. This motivates the search for a truly directed analogue of tree-width that can distinguish cases in which edge-orientations can make problems tractable.

It is not at all obvious how to obtain such an analogue. In fact this question has been an active area of research since the 1990s and it has stimulated the research community to define numerous directed analogues of tree-width [68]. All of these digraph width measures (which we shall call '*tree-width-inspired*' measures, following the terminology in [68]) were defined by generalizing a characterization of tree-width in terms of cops-robber games; they include *directed tree-width* [63], *DAG-width* [13], *Kelly-width* [59] and *D-width* [91] (see Figure 2.1 for an illustration of the relationships between these measures). Unfortunately, despite being very useful in some cases, in general all tree-width-inspired measures face considerable algorithmic shortcomings. For example, they are all bounded on the class of DAGs, a class for which some natural problems remain NP-complete (e.g. directed Max-Cut [69]). In fact, unless NP $\subseteq$ **P/poly**, certain algorithmic shortcomings are unavoidable by any digraph width measure which shares certain properties with any tree-width inspired

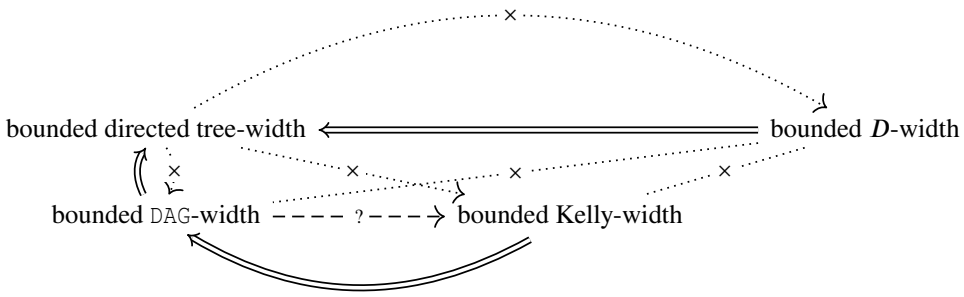


Figure 2.1: Implications of boundedness between tree-width-inspired measures (adapted from a diagram of Kreutzer and Kwon [68]). Legend: dotted (be they directed or bi-directed) arrows and marked with ×, indicate the failure of the relevant implications. The dashed arrow labeled with a question mark indicates that the existence of this implication is an open problem. All other arrows indicate implications.

measure (in particular there cannot be a Courcelle-like theorem parameterized by any one of these digraph width measures) [47].

Another notable class of digraph width measures is that of '*rank-width-inspired*' measures [68]. Rank-width is a generalization of tree-width which is bounded on some denser graph classes. For example, while every class of bounded tree-width has bounded rank-width, cliques and complete bipartite graphs have bounded rank-width (but unbounded tree-width). There are several directed analogues of rank-width, including *clique-width* (defined on digraphs from the start [27]), *NLC-width* [54] and *bi-cut-rank-width* [64]. These measures are unbounded on DAGs and they admit an algorithmic meta-theorem for problems definable in a more restricted fragment of logic [28]. However, this fragment of logic does not have enough expressive power to describe all the problems which are tractable on classes of bounded undirected tree-width. In fact the directed versions of the Hamilton Path and Max-Cut problems (both tractable on classes of bounded underlying tree-width) are **W**[**1**]-hard when parameterized by any tree-width or rank-width-inspired measure [45, 69].

Our contribution We introduce a new digraph width measure called *directed branch-width*. We obtain this measure by generalizing Gurski and Wanke's characterization of graph classes of bounded tree-width in terms of their line graphs [53]. We show that the Hamilton Path problem and the directed Max-Cut problem are in FPT parameterized by directed branch-width. This is particularly significant since both of these problems are **W**[**1**]-hard when parameterized by any tree-width-inspired or rank-width-inspired measure [45, 69]. More generally we show that there exists an FPT-time algorithm parameterized by directed branch-width for the model-checking problem on a restricted variant of the monadic second-order logic of graphs.

Chapter Outline In Section 2.2 we introduce the necessary directed graph-theoretic notions and we introduce the concepts of undirected branch-width and rank-width. In Section 2.3 we define directed branch-width

and we show that a class of digraphs has bounded directed branch-width if and only if its corresponding class of directed line-graphs has bounded bi-cut-rank-width. In Section 2.4 we study simple properties of directed branch-width and its relationship to existing digraph width-measures. In Section 2.5 we consider the algorithmic applications of directed branch-width. Finally we suggest future research directions in Section 2.6.

2.2 Background

For any vertex x in a digraph D, we shall write $N_D^+(x)$ and $N_D^-(x)$ to mean the sets $\{y \in V(D) : \overrightarrow{xy} \in E(D)\}$ and $\{w \in V(D) : \overrightarrow{wx} \in E(D)\}$ respectively of *out-neighbors* and *in-neighbors* of x in D. Furthermore, we shall denote by $N_D(x)$ the set $N_D^+(x) \cup N_D^-(x)$ of *neighbors* of x. Note that, if the digraph D is clear from context, then we shall drop the subscripts and simply write $N(x)$, $N^+(x)$ and $N^-(x)$. Given a directed graph D, we denote by $u(D)$ the *underlying undirected graph* of D which is obtained by making all of the edges of D undirected and removing parallel edges that might appear. We say that a subset A of vertices is *complete to* another vertex-subset B (disjoint from A) if every vertex of A is adjacent to every vertex of B.

The *line-graph* of a graph G was introduced by Whitney [94] and is the graph defined as $(E(G), \{ef : e, f \in E(G) \text{ s.t. } e \cap f \neq \emptyset\})$. We introduce the concept of a *directed line-graph* as a directed analogue of Whitney's definition. Just as the line-graph of an undirected graph encodes which pairs of edges can occur in succession in a path, the directed line-graph of a digraph D encodes which edge-pairs can occur in succession in a directed path.

Definition 2.2.1. The *directed line-graph* $\vec{L}(D)$ of a digraph D is the digraph

$$\vec{L}(D) := \left(E(D), \left\{\overrightarrow{ef} : \exists \{w, x, y\} \subseteq V(D) \text{ such that } e = \overrightarrow{wx} \text{ and } f = \overrightarrow{xy}\right\}\right).$$

Now we define the directed vertex and edge separators corresponding to edge and vertex-partitions respectively.

Definition 2.2.2. Let D be a directed graph, let the *ordered* pairs $(V(D) \setminus A, A)$ and $(E(D) \setminus B, B)$ be respectively partitions of the vertices of D and of the edges of D. We call the sets

$$S_A^E := \{\overrightarrow{xy} \in E(D) : x \in V(D) \setminus A \text{ and } y \in A\} \text{ and}$$

$$S_B^V := \{y \in V(D) : \exists x, z \in V(D) \text{ with } \overrightarrow{xy} \in E(D) \setminus B \text{ and } \overrightarrow{yz} \in B\}$$

the *edge separator* and *vertex separator corresponding to* $(V(D) \setminus A, A)$ *and* $(E(D) \setminus B, B)$ respectively. The *directed order of an edge (or vertex) partition* is the number of elements contained in the vertex (or edge) separator associated with that partition (for example the order of $(E(D) \setminus B, B)$ is $|S_B^V|$).

Every edge of a tree T partitions the set $\ell(T)$ *of the leaves of* T into two sets. Given an edge xy in a tree T and letting Y be the set of all leaves found in the connected component of $T - xy$ which contains the vertex y, we call the partition $\{\ell(T) \setminus Y, Y\}$ the *leaf-partition* of T corresponding to xy.

For any graph-theoretic notation not defined here, we refer the reader to [30].

2.2.1 Tree-width-inspired measures.

As we already mentioned in Chapter 1, there are several equivalent definitions of tree-width. One unified way of defining both undirected tree-width and all of the known tree-width-inspired measures for digraphs [68] is via a family of pursuit-evasion games called *cops-robber games* which are played on either directed or undirected graphs. These games are played on any graph (resp. digraph) with two players: the *Sheriff* and the *Villain*. The Sheriff controls a set of cops which each occupy a single vertex and the Villain controls a single robber which also occupies a single vertex. The Sheriff wins if a cop is ever placed on a vertex occupied by the robber and looses otherwise. Different cops-robber games can be defined by varying how the cops and the robbers can be moved and by the information available to each player. Depending on the game variant, undirected tree-width and each tree-width-inspired width measure can be defined as the number of cops needed to guarantee a win for the Sheriff [29, 68]. For a formal description of these games, see Section 2.4.3.

2.2.2 Layouts: branch-width and rank-width.

A notion closely related to tree-width, which we use throughout this thesis, is that of *branch-width*.

Definition 2.2.3 ([87])**.** A *branch decomposition* of a graph G is a pair (T, τ) where T is a tree of maximum degree three and τ is a bijection from the leaves of T to $E(G)$. The *order of an edge* e of T is the number of vertices v of G such that there are leaves t_1, t_2 in T in different components of $T - e$, with $\tau(t_1), \tau(t_2)$ both incident with v. The *width* of (T, τ) is the maximum order of the edges of T. The *branch-width* of G (written $\mathbf{bw}(G)$) is the minimum possible width of any branch-decomposition of G.

It is known that a class of undirected graphs has bounded tree-width if and only if it has bounded branch-width.

Theorem 2.2.4 ([87]). *For any graph G,* $\mathbf{bw}(G) \leq \mathbf{tw}(G)+1 \leq \max\{1, 3\,\mathbf{bw}(G)/2\}$.

We note that, if a graph has fewer than two edges, then it has branch-width zero; furthermore, all stars have branch-width at most 1 and all forests have branch-width at most 2 (this follows by Theorem 2.2.4 since forests have tree-width at most 1; the bound is tight since the three-edge path has branch-width 2).

A branch-decomposition is a special case of the more general concept of a *layout of a symmetric function* which can be applied even to structures which are not graphs. Given sets A and B, a function $f : 2^A \to B$ is *symmetric* if, for any $X \subseteq A$ we have $f(X) = f(A \setminus X)$.

Definition 2.2.5. Let U be a finite set and let $f : 2^U \to \mathbb{Z}$ be a symmetric function. We say that the pair (T, β) is a *layout of f on U* if T is a tree of maximum degree three and β is a bijection from the leaves of T to the elements of U.

Let xy be an edge in T and let $\{\ell(T) \setminus Y, Y\}$ be the leaf-partition associated with xy. We define the *order of the edge xy* (denoted $||xy||_f$) to be the value of $f(\beta(Y))$. The *width* of (T, τ) is the maximum order of all edges of T. The *layout-f-width of U* is defined as the minimum possible width of any layout of f on U.

Using the concept of a layout, we can now give an alternative definition of the branch-width of a graph G: the branch-width of G is the layout-f-width of $E(G)$ where f maps any edge subset X of G to the number of vertices incident with both an edge in $E(G) \setminus X$ and an edge in X.

Whereas branch-width can be thought of as a global measure of vertex-connectivity, *rank-width* (originally introduced by Bouchet [16][1]) is intuitively a measure of global neighborhood similarity. All classes of bounded branch-width have also have bounded rank-width, but the converse is not true, for example the rank-width of any clique is 1. Given a matrix M over $\mathbb{GF}(2)$ whose rows and columns are indexed by some set X, we denote by $\mathbf{rk}(M)$ the rank of M and, for $Y \subseteq X$, we denote by $\mathbf{rk}(M[X \setminus Y, Y])$ the rank of the sub-matrix of M given only by the rows indexed by elements of $X \setminus Y$ and the columns indexed by the elements of Y. If M is the adjacency matrix of a graph G and X is a vertex subset of G, then $\mathbf{rk}(M[V(G) \setminus X, X])$ describes the number of different kinds of edge-interactions between vertices in $V(G) \setminus X$ and those in X; layouts can be used to turn this local notion into a global one by defining rank-width as follows [16, 83].

[1] note that Bouchet does not call it "rank-width"; the name is instead due to Oum and Seymour [83]

Definition 2.2.6 ([16,83]). Let M be the adjacency matrix over $\mathbb{GF}(2)$ of a graph G. The *rank-width* $\mathbf{rw}(G)$ of G is the layout-f-width of $V(G)$ where we define f as the mapping $f : X \mapsto \mathbf{rk}(M[V(G) \setminus X, X])$.

Kanté and Rao generalized rank-width to digraphs by introducing the concept of a *bi-cut-rank decomposition* [64]. They did so by considering layouts of a function which takes into account the two possible directions in which edges can point in the directed edge separator associated with a vertex partition.

Definition 2.2.7 ([64]). Given a digraph D and its adjacency matrix M over $\mathbb{GF}(2)$, let $f(M[X])$ be the function defined as $f(M[X]) = \mathbf{rk}(M[V(D) \setminus X, X]) + \mathbf{rk}(M[X, V(D) \setminus X])$. The *bi-cut-rank-width* of a digraph D (denoted $\mathbf{bcrk}(D)$) is the layout-f-width of $V(D)$.

For completeness we note that rank-width is closely related to a width-measure called *clique-width* (defined in [27]).

Theorem 2.2.8 ([54], [83]). *A class of undirected graphs has bounded clique-width if and only if has bounded rank-width.*

Gurski and Wanke showed that classes of bounded tree-width can be characterized via the rank-width of their line graphs [53]. (We note that this result was originally stated in terms of clique-width; by Theorem 2.2.8, the following formulation is equivalent.)

Theorem 2.2.9 ([53], [81]). *A class of undirected graphs has bounded tree-width if and only if the class of its line graphs has bounded rank-width.*

Theorem 2.2.9 is of particular importance since it can be seen as a definition of classes of bounded tree-width. In Section 2.3 we will prove a directed analogue of Theorem 2.2.9.

2.2.3 Meta-obstructions

The authors of [47] investigate the question of why there are no known digraph width measures that are as algorithmically successful as tree-width in terms of parameterizations for FPT algorithms. They prove that unless NP $\subseteq$ P/POLY, it is impossible to obtain an XP algorithm for the the $\mathbf{MSO}_1$-model-checking problem parameterized by any digraph width measure similar to tree-width-inspired measures. Their result (Theorem 2.2.10) mentions a digraph containment relation called *directed topological minor relation* [47] which we define formally in Section 2.4.

Theorem 2.2.10 ([47]). *A digraph width-measure δ is said to be tree-width bounding if there exists a computable function b such that for every digraph D with $\delta(D) \leq k$, we have $\mathbf{tw}(u(D)) \leq b(k)$. If δ is a digraph width measure which is:*

- *not tree-width-bounding and*
- *closed under taking directed topological minors,*

then, unless NP $\subseteq$ P /POLY, *there is no* XP *algorithm for the* $\mathbf{MSO}_1$*-model-checking problem parameterized by δ.*

We point out that Theorem 2.2.10 excludes the existence of an XP algorithm (defined in Chapter 1): this is a much stronger claim than excluding fixed-parameter-tractability.

2.3 Directed line graphs of bounded rank-width.

In this section we introduce our new directed width measure, directed branch-width, and use it to generalize Theorem 2.2.9 to digraphs. Specifically we will prove that a class $\mathcal{C}$ of digraphs has bounded directed branch-width if and only if the class $\vec{L}(\mathcal{C})$ of directed line graphs of $\mathcal{C}$ has bounded bi-cut-rank-width. Note that all rank-width-inspired measures [64] are bounded on the same graph classes as bi-cut-rank-width [68], so this result can also be stated in terms of any one of the rank-width-inspired measures.

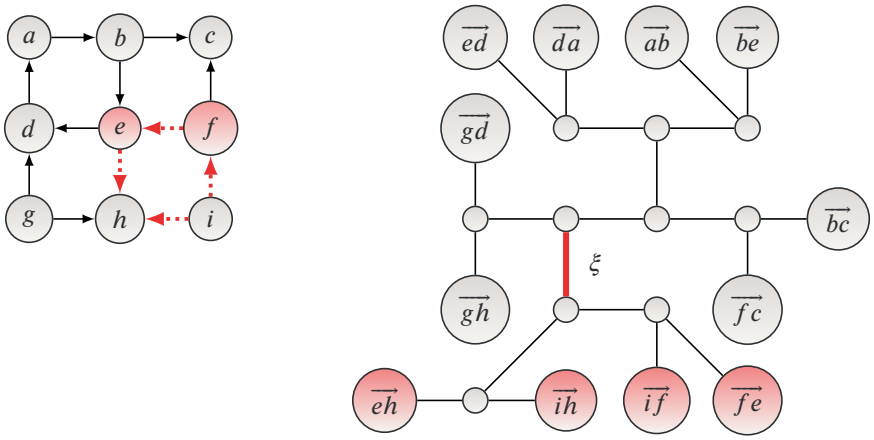


Figure 2.2: An orientation D of a (3×3)-grid (left) and a directed branch decompositions of this grid (right). Letting $X = \{\overrightarrow{eh}, \overrightarrow{ih}, \overrightarrow{if}, \overrightarrow{fe}\}$, the edge ξ is associated with the edge partitions $(E(G)\setminus X, X)$ and $(X, E(G)\setminus X)$. These partitions are themselves respectively associated with the directed vertex separators $\{e\}$ and $\{e, f\}$.

The definition of directed branch-width relies on the notion of directed vertex-separator. Recall from

Definition 2.2.2 that, for any edge-subset X of a digraph D, S^V_X and $S^V_{E(D)\setminus X}$ denote the directed vertex-separators corresponding to $(E(D)\setminus X, X)$ and $(X, E(D)\setminus X)$ respectively.

Definition 2.3.1 (Directed branch-width)**.** For any digraph D, let f_D be the function $f_D : 2^{E(D)} \to \mathbb{N}$ defined as $f_D(X) = |S^V_X \cup S^V_{E(D)\setminus X}|$. We call any layout of f_D on $E(D)$ a *directed branch decomposition* of D (see Figure 2.2). The *directed branch-width* of D, denoted $\mathbf{dbw}(D)$, is the layout-f_D-width of $E(D)$.

The rest of this section is devoted to the proof of following result, which generalizes Theorem 2.2.9 to directed branch-width and bi-cut-rank-width.

Theorem 2.3.2. *A class $\mathcal{C}$ of digraphs without parallel edges has bounded directed branch-width if and only if the class of directed line-graphs of $\mathcal{C}$ has bounded bi-cut-rank-width. Specifically, for any digraph D without parallel edges, we have*

$$\mathbf{bcrk}(\vec{L}(D))/2 - 1 \leq \mathbf{dbw}(D) \leq 8(1+2^{\mathbf{bcrk}(\vec{L}(D))}).$$

We shall first show (Lemma 2.3.4) that if a class $\mathcal{C}$ has bounded directed branch-width then $\vec{L}(\mathcal{C})$ has bounded bi-cut-rank-width. We shall show the converse of this statement in Lemma 2.3.7. To prove these results we shall make use of an auxiliary notion of consistency between a labeling function and a vertex-partition.

Definition 2.3.3. Let S be a set, (A, B) be a vertex partition in a digraph D and let $\lambda_A : A \to S$ and $\lambda_B : B \to S$ be functions. We say that (A, B) *is* (λ_A, λ_B)*-consistent* if

- given vertices a_1, a_2 in A with $\lambda_A(a_1) = \lambda_A(a_2)$ and a vertex b in B we have that $\overrightarrow{a_1 b} \in E(D)$ if and only if $\overrightarrow{a_2 b} \in E(D)$ and
- given vertices b_1, b_2 in B with $\lambda_B(b_1) = \lambda_B(b_2)$ and a vertex a in A we have that $\overrightarrow{a b_1} \in E(D)$ if and only if $\overrightarrow{a b_2} \in E(D)$.

For clarity we emphasize that the (λ_A, λ_B)-consistency of a partition (A, B) only concerns the edges that emanate from A and enter B. Thus the fact that a vertex-partition (A, B) of a digraph D is (λ_A, λ_B)-consistent does not necessarily imply that (B, A) is also consistent with (λ_B, λ_A).

Oum showed that, if $L(G)$ is the undirected line-graph of an undirected graph G, then $\mathbf{rw}(L(G)) \leq \mathbf{bw}(G)$ [81]. We will now show a directed analogue of this result with a weaker bound: the bi-cut-rank-width of a directed line graph of some digraph D is bounded by a linear function of the directed branch-width of D.

Lemma 2.3.4. *Let D be a digraph; if* $\mathbf{dbw}(D) \leq k$, *then* $\mathbf{bcrk}(\vec{L}(D)) \leq 2(k+1)$.

Proof. Throughout, let M be the adjacency matrix of $\vec{L}(D)$ over $\mathbb{GF}(2)$ whose rows and columns are indexed by $E(D)$. Let (T, β) be a directed branch decomposition of D of width at most k.

Notice that since $E(D) = V(\vec{L}(D))$ and since the leaves of T correspond bijectively (via β) to the elements of $E(D)$, it follows that they are also in bijective correspondence with $V(\vec{L}(D))$ (again, witnessed by β). In particular, this means that (T, β) can be also seen as a bi-cut-rank-decomposition of $\vec{L}(D)$. Thus it suffices to show that, if $(E(D) \setminus X, X)$ is any edge partition of D corresponding to an edge of T, then the value of $\mathbf{rk}(M[E(D) \setminus X, X])$ is at most $k+1$ since then this would imply that, when viewed as a bi-cut-rank decomposition, the width of (T, β) is at most $2(k+1)$.

Now fix a vertex partition $(E(D) \setminus X, X)$ of $\vec{L}(D)$ (recall that edge partitions of D are vertex partitions of $\vec{L}(D)$) and suppose that it has directed order at most k in D. Furthermore, let $\lambda_1 : E(D) \setminus X \to S$ and $\lambda_2 : X \to S$ be labeling functions. Notice that, if $(E(D) \setminus X, X)$ is (λ_1, λ_2)-consistent, then it must be that any two rows of $M[V(\vec{L}(D)) \setminus X, X]$ indexed by vertices of $V(\vec{L}(D)) \setminus X$ which have the same label under λ_1 are identical and hence linearly dependent. In particular this implies that if a vertex partition $(E(D) \setminus X, X)$ of $\vec{L}(D)$ is (λ_1, λ_2)-consistent, then $\mathbf{rk}(M[E(D) \setminus X, X]) \leq |S|$ since there are at most $|S|$ possible labels.

It therefore remains only to construct labeling functions λ_1 and λ_2 mapping the edges of D to a set with at most $k+1$ elements and show the (λ_1, λ_2)-consistency of $(E(D) \setminus X, X)$.

Let $\{s_1, \ldots, s_r\}$ be the vertex separator associated with the partition $(E(D) \setminus X, X)$ (note that, since (T, β) has width at most k, we have $r \leq k$). Define $\lambda_1 : E(D) \setminus X \to [r] \cup \{0\}$ and $\lambda_2 : X \to [r] \cup \{0\}$ as:

$$\lambda_1(\overrightarrow{xy}) := \begin{cases} i \text{ if } y = s_i \\ 0 \text{ otherwise} \end{cases} \quad \text{and} \quad \lambda_2(\overrightarrow{xy}) := \begin{cases} i \text{ if } x = s_i \\ 0 \text{ otherwise.} \end{cases}$$

Note that an edge $\overrightarrow{xy}$ will have label zero if it is in $E(D) \setminus X$ and does not point towards an element of $\{s_1, \ldots, s_r\}$, or if it is an element of X and it does not emanate from an element of $\{s_1, \ldots, s_r\}$.

We claim that $(E(D) \setminus X, X)$ is a (λ_1, λ_2)-consistent vertex-partition in $\vec{L}(D)$ (again recall that edge-partitions in D are vertex-partitions in $\vec{L}(D)$). To see this, let e_1 and e_2 be two elements of $E(D) \setminus X$ with $\lambda_1(e_1) = \lambda_1(e_2) = i$. If $i = 0$, then neither e_1 nor e_2 points towards a vertex in $\{s_1, \ldots, s_r\}$. Thus, for all f in

X, neither $\overrightarrow{e_1 f}$ nor $\overrightarrow{e_2 f}$ is an edge in $\vec{L}(D)$. Otherwise, if $i \neq 0$, then e_1 and e_2 both point towards the vertex s_i. This means that, for any $f \in X$ and $j \in [2]$, there will be an edge $\overrightarrow{e_j f}$ in $\vec{L}(D)$ if and only if f points away from s_i. Thus $\overrightarrow{e_1 f} \in E(\vec{L}(D))$ if and only if $\overrightarrow{e_2 f} \in E(\vec{L}(D))$. A symmetric argument also shows that for $f_1, f_2 \in X$ and $e \in E(D) \setminus X$, if $\lambda_2(f_1) = \lambda_2(f_2)$, then $\overrightarrow{e f_1} \in E(\vec{L}(D))$ if and only if $\overrightarrow{e f_2} \in E(\vec{L}(D))$. The partition $(E(D) \setminus X, X)$ is therefore (λ_1, λ_2)-consistent and the result follows. ■

Our next goal is to prove the converse of Lemma 2.3.4: given a digraph D such that $\vec{L}(D)$ has bi-cut-rank-width at most k, we shall deduce a bound on the directed branch-width of D. We obtain this result by extending techniques introduced by Gurski and Wanke in [53] to the setting of digraphs. Once again we shall be using the concept of consistency with respect to a labeling function as an intermediate step in our results. To do so we shall first need an auxiliary result giving a bound on the number of labels needed in order to find labeling functions with which a vertex partition of rank k is consistent.

Lemma 2.3.5. *Let $(V(D) \setminus X, X)$ be a vertex partition of a digraph D and let M be the adjacency matrix of D over $\mathbb{GF}(2)$. If $\mathbf{rk}(M[V(D) \setminus X, X]) \leq k$, then there exists a set S with $|S| \leq 1 + 2^k$ and labeling functions $\lambda_1 : V(D) \setminus X \to S$ and $\lambda_2 : X \to S$ such that $(V(D) \setminus X, X)$ is (λ_1, λ_2)-consistent.*

Proof. Let A be the subset of all vertices in $V(D) \setminus X$ which have an outgoing edge to an element of X and let B be the set of all vertices in X which have an incoming edge from an element of $V(D) \setminus X$. We define two equivalence relations $\sim_A$ and $\sim_B$ on A and B as:

- for all x and y in A, $x \sim_A y$ if $N^+(x) \cap X = N^+(y) \cap X$,
- for all x and y in B, $x \sim_B y$ if $N^-(x) \cap V(D) \setminus X = N^-(y) \cap V(D) \setminus X$.

Let $\{a_1, \ldots, a_p\}$ and $\{b_1, \ldots, b_q\}$ be sets of representatives of the equivalence classes of $\sim_A$ and $\sim_B$ in A and B respectively. Now we define the labeling functions $\lambda_1 : V(D) \setminus X \to [p] \cup \{0\}$ and $\lambda_2 : X \to [q] \cup \{0\}$ as

$$\lambda_1(x) = \begin{cases} 0 & \text{if } x \notin A \\ i & \text{if } x \text{ is in the equivalence class represented by } a_i \text{ under } \sim_A \end{cases}$$

$$\lambda_2(y) = \begin{cases} 0 & \text{if } y \notin B \\ j & \text{if } y \text{ is in the equivalence class represented by } b_j \text{ under } \sim_B . \end{cases}$$

These definitions imply that two vertices in either $V(D) \setminus X$ or X have the same label under either λ_1 or λ_2 if and only if they share exactly the same out-neighbors in X or in-neighbors in $V(D) \setminus X$ respectively. Thus $(V(D) \setminus X, X)$ is (λ_1, λ_2)-consistent.

All that remains to be shown is that p and q are both at most 2^k. We will argue only for p since the argument is the same (replacing rows for columns and in-neighborhoods for out-neighborhoods) for the bound on q. To see that $p \leq 2^k$, note that two vertices in A have the same out-neighborhood in X if and only if the row-vectors that they index in $M[V(D) \setminus X, X]$ are identical. Since $M[V(D) \setminus X, X]$ has rank at most k, there are at most 2^k linear combinations with binary coefficients of the basis vectors of the row-space of $M[V(D) \setminus X, X]$. This concludes the proof since p is at most the number of out-neighborhoods in X. ■

Now, for some digraph D, suppose we have a partition $(V(\vec{L}(D)) \setminus X, X)$ of rank k. By Lemma 2.3.5 we know that there exist functions λ_1, λ_2 mapping vertices of $V(\vec{L}(D)) \setminus X$ and X to at most ℓ labels (for $\ell \leq 1 + 2^k$). Our next step is to bound the order of the edge-partition of D corresponding to $(V(\vec{L}(D)) \setminus X, X)$ by a function of ℓ. The following lemma is a generalization of part of the proof of [53, Theorem 9] to the setting of digraphs.

Lemma 2.3.6. *Let D be a digraph without parallel edges and $(E(D) \setminus X, X)$ an edge-partition of D. Let $\lambda_1 : E(D) \setminus X \to S$ and $\lambda_2 : X \to S$ be functions to some k-element set S. If $(E(D) \setminus X, X)$ is (λ_1, λ_2)-consistent in $\vec{L}(D)$, then $(E(D) \setminus X, X)$ has directed-order at most $4k$ in D.*

Proof. Let $\xi : S \to 2^S$ be the function mapping any label a in S to the set

$$\xi(a) := \{b \in S : \exists e_1 \in \lambda_1^{-1}(a) \text{ and } \exists e_2 \in \lambda_2^{-1}(b) \text{ with } \overrightarrow{e_1 e_2} \in E(\vec{L}(D))\}$$

of all labels in S associated by λ_2 to vertices of $\vec{L}(D)$ which are the endopints of a directed edge emanating from some vertex labeled a by λ_1 (for clarity, recall that $V(\vec{L}(D)) = E(D)$). For any label a in S, denote by D_a the subgraph of D with $V(D_a) = V(D)$ and edge-set

$$E(D_a) = \lambda_1^{-1}(a) \cup \bigcup_{b \in \xi(a)} \lambda_2^{-1}(b).$$

We claim that it will suffice to show that, for every $a \in S$, the edge-partition $(\lambda_1^{-1}(a), E(D_a) \setminus \lambda_1^{-1}(a))$ has directed-order at most 4 in D_a. To see this, recall that every element of $E(D) \setminus X$ is labeled by λ_1 and every element of X is labeled by λ_2. This means that, for any element q in the vertex-separator of D associated with

$(E(D) \setminus X, X)$, there is a label a such that q belongs to the separator associated with $(\lambda_1^{-1}(a), E(D_a) \setminus \lambda_1^{-1}(a))$. Thus, since there are at most k labels (i.e. k choices for a), showing that $|(\lambda_1^{-1}(a), E(D_a) \setminus \lambda_1^{-1}(a))| \leq 4$ for every $a \in S$ would also then imply that $|(E(D) \setminus X, X)| \leq 4k$.

Thus let a be any label in S and, for notational simplicity, denote by (A, B) the partition $(\lambda_1^{-1}(a), E(D_a) \setminus \lambda_1^{-1}(a))$. Recall that, since A and B are edge-sets in D, they are vertex-sets in $\vec{L}(D)$. Furthermore, since $(E(D) \setminus X, X)$ is (λ_1, λ_2)-consistent in $\vec{L}(D)$, it follows (by the definitions of consistency and of ξ) that A is complete to B in $\vec{L}(D)$. In particular, this implies that in D_a every element of A is incident with every element of B.

We will now show that the partition (A, B) of D_a has order at most 4. If A contains at most 2 edges, then they have at most 4 endpoints and the claim is trivially true. So suppose that A has at least 3 edges.

Consider first the case in which A has at least two non-incident edges $e = \overrightarrow{wx}$ and $f = \overrightarrow{yz}$. Since every edge in B is incident with every edge in A, we can deduce in particular that every edge of B must be incident with both an element of $\{w, x\}$ and an element of $\{y, z\}$. Thus, since e and f are not incident, no edge of B has an endpoint outside of $\{w, x, y, z\}$. We may therefore conclude that $(E(A), E(B))$ has order at most 4 in D_a.

Otherwise, if A does not have at least two non-incident edges and since it has no parallel edges, then the elements of A must form a star in D_a. Since every edge of B is incident with every edge of A and since A has more than two edges, it must be that every edge of B is incident with the center of the star formed by A. In fact, since we are assuming no parallel edges, this implies that every edge of B is incident *exclusively* with the center of this star. Thus (A, B) has order most 1. ■

We emphasize that the requirement in Lemma 2.3.6 for D to not have any parallel edges is in fact necessary. To see this, consider the case in which A is a star with all edges pointing towards the center and B is the digraph with $V(A) = V(B)$ obtained from A by reversing the directions of all the edges in A. In this case the order of (A, B) in the digraph $Q := (V(A), E(A) \cup E(B))$ (which has parallel edges) would be $|A|$ (which equals $|B|$).

We now use Lemmas 2.3.5 and 2.3.6 to bound the directed branch-width of a digraph D in terms of the bi-cut-rank-width of its directed line-graph.

Lemma 2.3.7. *If D is a digraph without parallel edges with $\mathbf{bcrk}(\vec{L}(D)) \leq k$, then $\mathbf{dbw}(D) \leq 8(1 + 2^k)$.*

Proof. Let (T, β) be a minimum-width bi-cut-rank-decomposition of $\vec{L}(D)$ and let M be the adjacency matrix

of $\vec{L}(D)$. Since $V(\vec{L}(D)) = E(D)$, we know that (T, β) is also a directed branch-decomposition of D. Now let e be any edge in T and let $(E(D \setminus X), X)$ be the edge-partition of D associated with this edge.

Since $\mathbf{bcrk}(\vec{L}(D)) \leq k$, we know that $M[E(D) \setminus X, X]$ has rank at most k. By Lemma 2.3.5 we know that there exist vertex-labeling functions λ_1 and λ_2 such that every element of $V(\vec{L}(D))$ (i.e. $E(D)$) is mapped to one of at most $1+2^k$ labels and such that $(E(D \setminus X), X)$ is (λ_1, λ_2)-consistent. By Lemma 2.3.6 the (λ_1, λ_2)-consistency of $(E(D \setminus X), X)$ in $\vec{L}(D)$ implies that the directed order of $(E(D \setminus X), X)$ in D is at most $4(1+2^k)$. The result follows since the order of e in T is the cardinality of the union of the directed separators corresponding to $(E(D \setminus X), X)$ and $(X, E(D \setminus X))$. ■

Theorem 2.3.2 now follows immediately from Lemmas 2.3.4 and 2.3.7.

2.4 Properties of Directed branch-width.

Having introduced directed branch-width in the previous section, here we study some of its properties. In Section 2.4.1 we will show that classes of bounded directed branch-width need not have bounded underlying undirected branch-width. Despite this result, we will obtain some relationships between undirected and directed branch-width which will be useful for our algorithmic applications in Section 2.5. In Section 2.4.2 we study butterfly minors and directed topological minors (two directed analogues of the minor relation) and we will show that the directed branch-width is monotone under the first but not under the second. Finally in Section 2.4.3 we shall show that classes of digraphs that have bounded width with respect to any tree-width or rank-width-inspired measure need not have bounded directed branch-width.

2.4.1 Relationship to undirected branch-width.

Here we compare the directed branch-width of a digraph and the branch-width of its underlying undirected graph. We will prove that there exist digraph classes of bounded directed branch-width and unbounded underlying undirected branch-width. However, despite this result, we will find two ways of relating directed and undirected branch-width. First we will show that, for any digraph D, the difference between the branch-width of $u(D)$ and the directed branch-width of D is at most the number of sources and in sinks in D. Second we will show that, given any digraph D, we can obtain, by modifying D only at sources and sinks, a digraph H such that $\mathbf{dbw}(D) = \mathbf{bw}(u(H))$. All of these results will be important for our algorithmic applications in Section 2.5.

Towards proving our results here and those in Section 2.4.2 about butterfly and directed topological minors, we observe that directed branch-width is subgraph-closed.

Lemma 2.4.1. *If H is a subgraph of a digraph D, then* $\mathbf{dbw}(H) \leq \mathbf{dbw}(G)$.

Proof. Let (T, β) be a directed branch decomposition of D. Let T' be the inclusion-wise minimal subtree of T containing the leaves in the set $\{\ell \text{ leaf in } T : \beta(\ell) \in E(H)\}$. Notice that, letting β' be the restriction of β to T', the pair (T', β') is a directed branch decomposition of H. Now take any partition $(E(D) \setminus X, X)$ of D induced by an edge which is contained in both T and T'. Since its order when restricted to the edge set of H (i.e. the partition $(E(H) \setminus X, E(H) \cap X)$) is at most that of $(E(D) \setminus X, X)$, the width of (T', β') is at most that of (T, β). ■

Now we introduce some more notation which will be convenient for the next results. Let (T, β) be a directed branch decomposition of width k of a digraph D. By the definition of (T, β), every edge e of T is associated with two edge-partitions $(E(D) \setminus Y, Y)$ and $(Y, E(D) \setminus Y)$ of order at most k. We associate to every *internal edge* e of T three sets: X_e, $X_{e,Y}$, $X_{e,E(D)\setminus Y}$. The sets $X_{e,Y}$ and $X_{e,E(D)\setminus Y}$ are the *directed separators* at e associated with the partitions $(E(D) \setminus Y, Y)$ and $(Y, E(D) \setminus Y)$ respectively; we denote the set $X_{e,Y} \cup X_{e,E(D)\setminus Y}$ as X_e and we call it the *bidirected separator* at e. We point out that, using with this notation, the width of a directed branch-decomposition (T, β) is $\max_{e \in E(T)} |X_e|$.

In contrast to undirected graphs, in digraphs it is possible to have vertices of high degree (e.g. sources or sinks) which never appear as internal vertices in directed paths. The following result shows that such vertices never appear in bidirected separators of directed branch decompositions.

Lemma 2.4.2. *Let e and x be respectively an edge and a vertex in a digraph J.*

1. *If x never appears as an internal vertex of a directed path in J, then x will never appear in a bidirected separator of any directed branch-decomposition of J.*

2. *If e never appears in a directed path with at least two edges in J, then* $\mathbf{dbw}(J - e) = \mathbf{dbw}(J)$.

Proof. Note that (1) follows immediately from the definition of directed branch-width; furthermore (1) implies (2) since the endpoints of e satisfy the conditions of (1). ■

This result immediately implies that there is an infinite set of digraphs given by acyclic orientations of the square grids which has bounded directed branch-width. To see this, let Γ be the *set of all square grid-graphs*.

Define $\vec{\Gamma}$ to be the class of digraphs obtained by orienting the edges of every graph G in Γ as follows: take a proper two-coloring of G with colors white and black and then orient every edge towards its black endpoint.

Corollary 2.4.3. *Every graph in $\vec{\Gamma}$ has directed branch-width 0.*

Proof. Every vertex of any element D of $\vec{\Gamma}$ is either a source or a sink. It therefore follows that every bidirected separator of any directed branch-decomposition of D is empty (by Lemma 2.4.2) and hence that $\mathbf{dbw}(D) = 0$. ■

Now we show that the boundedness of directed branch-width does not imply boundedness of underlying undirected branch-width.

Theorem 2.4.4. *There does not exist any function $f : \mathbb{N} \to \mathbb{N}$ such that, for every digraph D, $\mathbf{bw}(u(D)) \leq f(\mathbf{dbw}(D))$.*

Proof. By Corollary 2.4.3, there is an infinite set $\vec{\Gamma}$ of acyclic orientations of the square grids which has directed branch-width zero. However, the $(n \times n)$-grid has tree-width n [30] and hence it has branch-width at least $2n/3$ (by Theorem 2.2.4). ■

In a similar vein to Lemma 2.4.2, we can show show that directed branch-width is invariant under a form of identifications of sources or sinks which we define now.

Definition 2.4.5. Two vertices x and y of a digraph D are *source/sink-identifiable* if they are either both sources or both sinks. We say that a digraph H is *equivalent to D under source-sink identification* if it can be obtained from D via a sequence of identifications of pairs of source/sink-identifiable vertices. We say that a graph D' is a *source-sink-split* of a digraph D if every source and every sink in D' has degree exactly 1 and if D is equivalent to D' under source-sink identification.

The next result shows that directed branch-width is invariant under source-sink identifications. Note that, for undirected graphs, where the natural analogue of a source or a sink is a pendant vertex, this does not hold. In fact, the branch-width of an undirected graph can be increased arbitrarily by repeated identification of pendant vertices.

Lemma 2.4.6. *Let H and D be two digraphs. If H is equivalent to D under source-sink identification, then $\mathbf{dbw}(H) = \mathbf{dbw}(D)$.*

Proof. Let x and y be two source/sink-identifiable vertices in D and note that it is sufficient to consider the case in which H is obtained from D by identifying x and y into a vertex γ.

Let (T, β) be a directed branch decomposition of D and define the map $\omega : E(D) \to E(H)$ as

$$\omega(uv) = \begin{cases} \overrightarrow{\gamma v} & \text{if } u \in \{x, y\} \\ \overrightarrow{u\gamma} & \text{if } v \in \{x, y\} \\ uv & \text{otherwise} \end{cases}$$

Clearly the map $\xi := \omega \circ \beta$ from the leaves of T to $E(H)$ is surjective; however, it would fail to be injective if x and y have a common neighbor. In this case, we can simply remove some duplicate leaves (i.e. leaves mapped by ξ to the same edge) in T so that we can ensure ξ is injective. We shall assume that we have done so to (T, ξ) if required.

Since x and y are st-identifiable, they never appear as internal vertices of a directed path. Thus, by Lemma 2.4.2, it follows that neither x nor y will ever appear in any bidirected separator of (T, β). Furthermore, we know that γ must also be either a source or a sink since identifying two sources or two sinks yields respectively a source or a sink. Thus γ will also never appear in any bidirected separator of (T, ξ) (again by Lemma 2.4.2). In particular, letting X be any edge-subset of D, this implies that the order of $(E(D) \setminus X, X)$ in G is the same as that of $(E(H) \setminus \omega(X), \omega(X))$ in H. By the definition of ξ this implies that the width of (T, β) is the same as that of (T, ξ). ■

The following result characterizes bi-directed vertex-separators in a digraph D in terms of vertex separators in $u(D)$ and the set of sources and sinks in D.

Lemma 2.4.7. *Let $(E(D) \setminus X, X)$ be an edge-partition of D and let S_1 and S_2 be the directed separators associated with $(E(D) \setminus X, X)$ and $(X, E(D) \setminus X)$ respectively. Let U be the set of all vertices incident both with an edge in $E(u(D)) \setminus X$ and with an edge in X in the underlying undirected graph $u(D)$. If S is the set of all sources and all sinks in D, then $S_1 \cup S_2 = U \setminus S$.*

Proof. Consider any element x in $S_1 \cup S_2$. By the definition of S_1 and S_2, there must be two edges $e \in E(D) \setminus X$ and $f \in X$ such that either $e = \overrightarrow{wx}$ and $f = \overrightarrow{xy}$ or such that $e = \overrightarrow{xw}$ and $f = \overrightarrow{yx}$. Either way, this implies that x is neither a source nor a sink and that $x \in U$ (since x is incident with both e and f in $u(D)$). Thus we have shown that $S_1 \cup S_2 \subseteq U \setminus S$.

Now consider any element $y \in U \setminus (S_1 \cup S_2)$. Since y is in U, we know that it is incident with at least two edges; so let g be one such edge. If y is the head of g, then we must have that every other edge incident

with y has y as its head (since otherwise y would be in $S_1 \cup S_2$). Similarly, if y is the tail of g, then every other edge incident with y must have y as its tail. These observations imply that y is either a source or a sink (i.e. an element of S). Thus $S_1 \cup S_2 = U \setminus S$. ■

Via Lemma 2.4.7 we now find a relationship between directed branch-width of a digraph D and the undirected branch-width of its underlying undirected graph which depends on the number of sources and sinks of D.

Corollary 2.4.8. *Let D be a digraph; if S is the set of all sources and sinks in D, then* $\mathbf{bw}(u(D)) - |S| \leq \mathbf{dbw}(D) \leq \mathbf{bw}(u(D))$.

Proof. Consider a directed branch-decomposition (T, β) of D. Letting $\phi : E(D) \to E(u(D))$ be the bijection mapping every directed edge to its undirected counterpart, note that $(T, \phi \circ \beta)$ is an undirected branch decomposition of $u(D)$. Furthermore, observe that, in this way, every undirected branch decomposition can be obtained from some directed branch decomposition and vice versa.

Let ε be any edge in T and suppose that it has order k in (T, β) and order ℓ in $(T, \phi \circ \beta)$. By Lemma 2.4.7, we know that $\ell - |S| \leq k \leq \ell$. Furthermore, this is true for any choice of ε. Thus, since ϕ establishes a bijective correspondence between the set of all directed branch decompositions of D and that of all undirected branch decompositions of $u(D)$, it follows that $\mathbf{bw}(u(D)) - |S| \leq \mathbf{dbw}(D) \leq \mathbf{bw}(u(D))$. ■

Note that, since any bidirected orientation D of an undirected graph G has no sources or sinks, we can apply Corollary 2.4.8 to deduce the following result.

Corollary 2.4.9. *If D is the digraph obtained by replacing every edge xy in an undirected graph G with the edges $\overrightarrow{xy}$ and $\overrightarrow{yx}$, then* $\mathbf{dbw}(D) = \mathbf{bw}(G)$.

We will now show that, given any digraph D, we can obtain, by modifying D only at sources and sinks, a digraph D' such that the directed branch-width of D equals the underlying undirected branch-width of D'.

Lemma 2.4.10. *If H is the source-sink-split of D, then* $\mathbf{bw}(u(H)) = \mathbf{dbw}(D)$.

Proof. By Lemma 2.4.7 we know that the vertex separator in $u(H)$ consists of the elements of the vertex separator it corresponds to in H as well as possibly some sources and/or sinks. By the definition of H, every source or sink in H has degree 1. Thus, for any vertex $x \in V(H)$ (recall $V(H) = V(u(H))$), x will appear as an internal vertex in a directed path in H if and only if it appears as an internal vertex in a path in $u(H)$. But then sources

and sinks of H (which, by the definition of H, have degree 1) will never appear in a vertex-separator in $u(H)$ (by the definition of undirected branch-width) and hence $\mathbf{bw}(u(H)) = \mathbf{dbw}(H)$. The result follows since, by Lemma 2.4.6, $\mathbf{dbw}(H) = \mathbf{dbw}(D)$. ■

2.4.2 Butterfly minors and directed topological minors.

Two directed analogues of the minor relation are *butterfly minors* and *directed topological minors*. The directed topological minor relation (introduced in [47]) is a less-restrictive variant of the better-known notion of a butterfly minor which was introduced in [63]. We show that although directed branch-width is not closed under topological minors, it is closed under butterfly minors.

Definition 2.4.11. The *digraph $D/\overrightarrow{xy}$ obtained by contracting an edge $\overrightarrow{xy}$ of a digraph D* is the digraph obtained from D by removing the edge $\overrightarrow{xy}$ and the vertices x and y and replacing these with a new vertex $v_{\overrightarrow{xy}}$ which has in-neighborhood and out-neighborhood equal to $N_D^-(x) \cup N_D^-(y)$ and $N_D^+(x) \cup N_D^+(y)$ respectively in $D/\overrightarrow{xy}$.

Both directed topological minors and butterfly minors are defined by taking subgraphs and contracting edges; what distinguishes them is which edges are deemed 'contractible'. Directed topological minors allow only the contraction of so-called 2-*contractible edges* (defined below). These are edges which, when contracted, do not introduce new directed paths between vertices of degree at least three.

Definition 2.4.12 ([47])**.** Let D be a digraph and let $V_3(D)$ denote the subset of vertices incident with at least 3 edges in D. An edge $a = \overrightarrow{xy}$ is 2-*contractible* in D if

- $\{x, y\} \not\subseteq V_3(D)$
- $\overrightarrow{yx} \in E(D)$ or there does **not** exist a pair of vertices (w, z) in $V_3(D)$, possibly $w = z$, such that x can reach w in $D - \overrightarrow{xy}$ and z can reach y in $D - \overrightarrow{xy}$.

Butterfly minors, on the other hand, only allow the contraction of *butterfly edges*.

Definition 2.4.13. Let $\overrightarrow{xy}$ be an edge in a digraph D. If x has out-degree 1 and y has in-degree 1, then we call $\overrightarrow{xy}$ a *butterfly edge*.

Note that the contraction of a butterfly edge never creates new directed paths that were not otherwise present. On the other hand, this might happen when contracting a 2-contractible edge (for example one joining a source to a sink).

Having defined these two kinds of edges, we can define directed topological minors and butterfly minors.

Definition 2.4.14 ([47, 63])**.** Let H and D be digraphs.

- H is a *directed topological minor* of D if H can be obtained from a subgraph of D via a sequence of contractions of 2-*contractible* edges. [47]
- H is a *butterfly-minor* of D if it can be obtained from a subgraph of D via sequence of contractions of *butterfly edges* [63].

The next theorem shows directed branch-width is not closed under directed topological minors. In contrast, we point out that all the tree-width-inspired measures are indeed closed under directed topological minors [47].

Theorem 2.4.15. *There does not exists any function $g : \mathbb{N} \to \mathbb{N}$ such that, for every directed topological minor H of a digraph J, we have* $\mathbf{dbw}(H) \leq g(\mathbf{dbw}(J))$.

Proof. We shall construct a set of graphs $\{D_n : n \in \mathbb{N}\}$ of bounded directed branch-width and show that there is a set $\{\Delta'_n : n \in \mathbb{N}\}$ of directed topological minors of elements of $\{D_n : n \in \mathbb{N}\}$ which has unbounded directed branch-width.

Define D_n by starting from the n-vertex directed cycle C_n and proceeding as follows:

- add n sources $s_1, \ldots, s_n$, each one adjacent to every vertex of C_n
- add the vertices $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ and, for each $i \in [n]$, add the edges $\overrightarrow{s_i a_i}$ and $\overrightarrow{b_i a_i}$ (see Figure 2.3).

We will now show that $\mathbf{dbw}(D_n) \leq 3$. Let D'_n be the source-sink-split of D_n and note that $u(D'_n)$ is a cycle with n pendant edges at each vertex together with some isolated edges (corresponding to the edges $\overrightarrow{s_i a_i}$ and $\overrightarrow{b_i a_i}$). Thus, by inspection, $u(D'_n)$ has tree-width 2 (since deleting a single vertex from the cycle in $u(D'_n)$ yields a forest). By Theorem 2.2.4, this implies that $\mathbf{bw}(u(D'_n)) \leq 3$ which, by Corollary 2.4.8, implies that $\mathbf{dbw}(D'_n) \leq 3$ and hence, by Lemma 2.4.6, that $\mathbf{dbw}(D_n) \leq 3$.

Now we will obtain a set $\{\Delta_n : n \in \mathbb{N}\}$ having unbounded directed branch-width such that Δ_n is a directed topological minor of D_n. Note that each edge $\overrightarrow{s_i a_i}$ is 2-contractible since, for each i, a_i has degree 2 and since the only vertex that can reach a_i in $D_n - \overrightarrow{s_i a_i}$ (namely b_i) has degree 1. Thus, by contracting every edge $\overrightarrow{s_i a_i}$, we construct the digraph Δ_n which is a directed topological minor of D_n. Now consider the digraph Δ'_n obtained

from Δ_n by identifying all of the sources $b_1, \dots, b_n$ into a single source b (see Figure 2.3). By Lemma 2.4.6 we know that $\mathbf{dbw}(\Delta'_n) = \mathbf{dbw}(\Delta_n)$ so we shall conclude the proof by showing that $\{\Delta'_n : n \in \mathbb{N}\}$ has unbounded directed branch-width. Since $u(\Delta'_n)$ contains the balanced complete bipartite graph $K^{n,n}$ as a subgraph and since $\mathbf{tw}(K^{n,n}) = n$ [30], we deduce (via Theorem 2.2.4) that

$$\mathbf{bw}(u(\Delta'_n)) \geq 2(\mathbf{tw}(K^{n,n})/3 - 1) = 2n/3 - 2.$$

Finally, since Δ'_n has at exactly one source and no sinks, we deduce by Corollary 2.4.8 that

$$\mathbf{dbw}(\Delta'_n) \geq \mathbf{bw}(u(\Delta'_n)) - 1 \geq 2n/3 - 3.$$

Thus we conclude that $\{D_n : n \in \mathbb{N}\}$ has bounded directed branch-width and $\{\Delta'_n : n \in \mathbb{N}\}$ has unbounded directed branch-width as required. ■

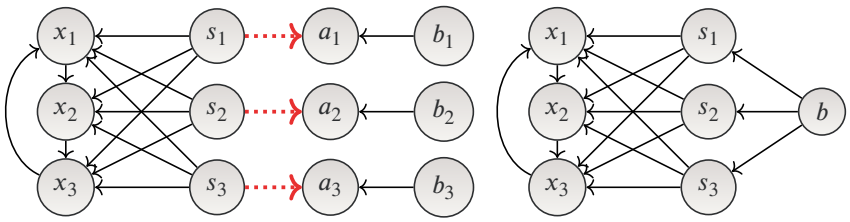


Figure 2.3: **Left**: the graph D_3 defined in the proof of Theorem 2.4.15 (the relevant 2-contractible edges are drawn red and dotted). **Right**: the graph Δ'_3.

Now we turn our attention to butterfly minors and show that directed branch-width is a butterfly-minor-closed parameter.

Theorem 2.4.16. *If H is a butterfly minor of a digraph D, then* $\mathbf{dbw}(H) \leq \mathbf{dbw}(D)$.

Proof. Since directed branch-width is subgraph-closed (by Lemma 2.4.1) it suffices to show that it is not increased under contractions of butterlfy edges. We will show that, given any directed branch decomposition (T, β) of D, we can construct a directed branch decomposition of $D/_{\overrightarrow{xy}}$ of width at most that of (T, β).

Let $\overrightarrow{xy}$ be a butterfly edge in D and let ω be the vertex of $D/_{\overrightarrow{xy}}$ created by the contraction of $\overrightarrow{xy}$. Let

$\gamma : E(D) \setminus \{\overrightarrow{xy}\} \to E(D/_{\overrightarrow{xy}})$ be the bijection defined for any edge $\overrightarrow{st}$ in $E(D) \setminus \{\overrightarrow{xy}\}$ as:

$$\gamma(\overrightarrow{st}) := \begin{cases} \overrightarrow{s\omega} & \text{if } t = x \\ \overrightarrow{\omega t} & \text{if } s = y \\ \overrightarrow{st} & \text{otherwise.} \end{cases}$$

Note that γ is well-defined since x and y are respectively a sink and a source in $E(D) \setminus \{\overrightarrow{xy}\}$ (by the definition of butterfly edge).

We will use γ to construct a directed branch-decomposition (U, δ) of $D/_{\overrightarrow{xy}}$ from (T, β) as follows. Let U be the inclusion-wise minimal subtree of T connecting all leaves of T which are not mapped to $\overrightarrow{xy}$ by β. Let $\delta := \gamma \circ (\beta|_{\ell(U)})$ where $\beta|_{\ell(U)}$ denotes the restriction of β to the set $\ell(U)$ of leaves of U (note that, by the definition of γ, δ is a bijection between the leaves of U and the edges of $D/_{\overrightarrow{xy}}$).

Let e be an edge in U (and note that $e \in E(T)$ also). Let $X_e^{(T,\beta)}$ be the bidirected separator at e in (T, β) and let $X_e^{(U,\delta)}$ be the bidirected separator at e in (U, δ). By the definition of (U, δ), we have that

$$X_e^{(U,\delta)} = \begin{cases} \{\omega\} \cup (X_e^{(T,\beta)} \setminus \{x, y\}) & \text{if } \{x, y\} \cap X_e^{(T,\beta)} \neq \emptyset \\ X_e^{(T,\beta)} & \text{otherwise} \end{cases}$$

and hence that $|X_e^{(U,\delta)}| \leq |X_e^{(T,\beta)}|$. Since this is true for any edge e, it implies that $\mathbf{dbw}(D/_{\overrightarrow{xy}}) \leq \mathbf{dbw}(D)$ as required. ■

2.4.3 Comparison to other digraph width measures.

All of the "tree-width-inspired" width measures (such as *directed tree-width*, *DAG-width*, *D-width* and *Kelly-width*) are bounded on the class of all DAGs [68]. In contrast we show that the class of all DAGs has *unbounded* directed branch-width.

Theorem 2.4.17. *For any $n \in \mathbb{N}$, there exists a directed acyclic graph D_n with $\mathbf{dbw}(D_n) \geq n$.*

Proof. Let $\kappa = 3(n+2)/2 - 1$ and D_n be the acyclic digraph obtained by orienting all edges of the $\kappa \times \kappa$ grid

north-east, i.e. the digraph

$$D_n := \left([\kappa]^2,\ \{\overrightarrow{(a,b)(c,d)} : (a = c \text{ and } d = b+1) \text{ or } (b = d \text{ and } c = a+1)\}\right).$$

Recall that, since $u(D_n)$ is a square grid, it has tree-width $\kappa = 3(n+2)/2 - 1$ [30]. Thus we can invoke Theorem 2.2.4 to deduce that $\mathbf{bw}(u(D_n)) \geq n+2$. Since D_n has exactly one source and one sink (the south-west and north-east corners), we know (by Corollary 2.4.8) that $\mathbf{dbw}(D_n) \geq \mathbf{bw}(u(D_n)) - 2$, so the result follows. ■

Recall that each tree-width-inspired measure (and tree-width itself) can be defined via some variant of a *cops-robber game* played on either a digraph or an undirected graph respectively. Using Theorem 2.4.17, we will show that none of these games can be used to characterize directed branch-width.

The cops-robber variants that can be used to define the tree-width-inspired measures all have the following common structure. The Sheriff can chose to add new cops to the board or move some cops that have been previously placed while the Villain can only move their piece. Each round follows this sequence of events: the Sheriff announces which cops they intend to move and where they intend to move them to, then they remove any cops that are involved in this move from the board; before the cops are placed back on the board, the Villain makes their move; finally the Sheriff completes their previously-announced move.

Playing on a (directed) graph D, the state of the game after round ρ is recorded by the pair $(C, r)_\rho$ where $C \subseteq V(D)$ is the set of positions of the cops and $r \in V(D)$ is the position of the robber. The initial state of the game is always of the form $(\emptyset, r)_0$ for some vertex r in D and the game terminates when a cop is placed on the vertex occupied by the robber. The Sheriff loses if the game never terminates and wins otherwise. The *width of a play at time* ρ is the maximum number of cops that were ever placed at any round up to and including ρ. We say that the Sheriff has a *k-winning strategy* on a (directed) graph D if they can always win on D with a play of width at most k.

The cops can be moved (or placed) freely from their position to any other vertex in the graph (this is often conceptualized as the cops being able to move "by helicopter"). Different game variants are distinguished by the knowledge of the Sheriff or the different ways in which a robber can be moved. The first distinction is whether or not the Sheriff knows the position of the robber; we call these games *visible* and *invisible-robber-game* respectively. The second important distinction emerges from how the Villain is allowed to move the robber. For the games used to characterize tree-width-inspired measures, the least restrictive movement pattern for the

robber is for it to be movable from position r in a (directed) graph G to any vertex r' which is reachable by a (directed) path from r in $G - C$, where C is the current vertex-set occupied by cops. This is called the *weak reachability game*. We note that we also allow the Villain to pass their turn without moving their piece (this is referred to as an *inert robber game* [68]).

The games we have described can be used to characterize both tree-width and any tree-width inspired measure. An undirected graph G has tree-width at most k if and only if the Sheriff has a $(k+1)$-winning-strategy for the visible cops-robbers game on G [29]. A similar statement can be made about any tree-width-inspired measure μ: there exists a cops-robber game Γ_μ such that the Sheriff has a k-winning-strategy for Γ_μ on a digraph D if and only if $\mu(D) \leq k$ [68].

Note that the most general game variant is the *invisible-robber inert-weak-reachability cops-robber game*. This follows since: (1) every strategy playable with a robber that cannot remain inert is also playable with one that can, (2) if the Sheriff can win the invisible-robber game, then they can also win in the visible version (by closing their eyes).

Corollary 2.4.18. *The Sheriff has a* 1*-winning strategy for the invisible-robber inert-weak-reachability cops-robber game on any directed acyclic graph.*

Proof. Let D be an acyclic digraph and let $v_1, \ldots, v_n$ be a topological ordering of its vertices. The Sheriff's strategy on D is to place one cop at v_1 and then move it at every round to the next vertex in the topological ordering. Note that, if the robber is moved at any round ρ from some vertex v_i to v_j, then it must be that $j \geq i$. Thus, since the cop will eventually reach v_n, the robber cannot escape indefinitely. ■

By Theorem 2.4.17, we know that the class of all DAGs has unbounded directed branch-width. Thus Corollary 2.4.18 implies the following result.

Corollary 2.4.19. *There is a family of digraphs of unbounded directed branch-width for which the Sheriff has a* 1*-winning strategy for the invisible-robber inert-weak-reachability cops-robber game.*

Undirected graph classes of bounded branch-width also have bounded rank-width [24, 82]. We will show that this is not true for the directed analogues of these measures: boundedness of directed branch-width does not imply boundedness of bi-cut-rank-width. To do so, we first recall a known result about the rank-width of grids.

Theorem 2.4.20 ([61]). *The class of all undirected grids has unbounded rank-width.*

For directed graphs, the bi-cut-rank-width version of Theorem 2.4.20 follows by the following theorem.

Theorem 2.4.21 ([54,68]). *For any digraph D, $\mathbf{bcrk}(D) \geq \mathbf{rw}(u(D))$.*

Thus Theorems 2.4.20 and 2.4.21 allow us to show that boundedness of directed branch-width does not imply boundedness of bi-cut-rank-width.

Theorem 2.4.22. *There does not exist a function $g : \mathbb{N} \to \mathbb{N}$ such that, for any digraph D, $\mathbf{bcrk}(D) \leq g(\mathbf{dbw}(D))$.*

Proof. By Corollary 2.4.3, there is an infinite set $\vec{\Gamma}$ of digraphs given by acyclic orientations of the square grids such that $\vec{\Gamma}$ has bounded directed branch-width. However, by Theorems 2.4.20 and 2.4.21, we know that $\vec{\Gamma}$ has unbounded bi-cut-rank-width. ■

2.5 Algorithmic aspects of directed branch-width.

In this section we will first show that directed branch-width is computable in FPT-time and then go on to demonstrate that many problems (such as Directed Hamilton Path and Directed MaxCut) are in FPT when parameterized by directed branch-width.

2.5.1 Computing directed branch-width

Here we will show that, despite being NP-complete, the problem of determining whether a digraph has branch-width at most k can be solved in linear time for any constant k.

We begin with a formal statement of the problem.

DIRECTED-BRANCH-WIDTH.

Input: a digraph D and a positive integer k.

Question: does D have directed branch-width at most k?

It is known that deciding whether the branch-width of an undirected graph is at most k is an NP-complete problem [92]. Since directed branch-width agrees with undirected branch-width on bi-directed orientations of

undirected graphs (Corollary 2.4.9), it follows that DIRECTED-BRANCH-WIDTH is NP-hard. Furthermore, given a directed branch decomposition (T, β) of a digraph D, one can check in polynomial time the order of any edge of T. Since T has $O(|E(D)|)$ edges, DIRECTED-BRANCH-WIDTH is in NP; thus we have shown the following result.

Theorem 2.5.1. DIRECTED-BRANCH-WIDTH *is* NP*-complete.*

A natural next question is whether computing the directed branch-width of a digraph D is in FPT parameterized by k. This is known for the undirected case.

Theorem 2.5.2 ([12, 15])**.** *There is a linear-time algorithm which decides for a fixed k whether an undirected graph has branch-width at most k.*

We can use Theorem 2.5.2 and Lemma 2.4.10 to show that deciding whether a digraph has directed branch-width at most k is in FPT parameterized by k.

Theorem 2.5.3. DIRECTED-BRANCH-WIDTH *is in* FPT *parameterized by k.*

Proof. Letting D' be the source-sink-split of D, we know (Lemma 2.4.10) that $\mathbf{bw}(u(D')) = \mathbf{dbw}(D)$. A source-sink-split can be obtained in polynomial time: for each source (or sink) x with $N(x) = \{y_1, \ldots, y_\eta\}$, replace x with sources (or sinks) $z_1, \ldots, z_\eta$ such that z_i is adjacent to y_i. Since $|V(D')|$ is $O(|V(D)|^2)$, we can use the FPT algorithm of Theorem 2.5.2 to determine whether the undirected branch-width of $u(D')$ is at most k and hence compute whether the directed branch-width of D is at most k in time linear in $|E(D')|$ (which equals $|E(D)|$). ■

2.5.2 Parameterizations by directed branch-width

Our definition of directed branch-width was motivated by the need for a digraph width-parameter that could be used to devise many FPT-time algorithms. As we mentioned in Section 2.1, despite their great success in some cases, the tree-width and rank-width-inspired measures still face considerable shortcomings when compared to the algorithmic power of undirected tree-width. In particular recall that the directed versions of the Hamilton Path and Max-Cut problems are $\mathbf{W}[\mathbf{1}]$-hard when parameterized by any rank-width or tree-width-inspired measure [45, 69].

D-HAMILTON-PATH

Input: a digraph D.

Question: does D contain a directed Hamiltonian Path?

D-MAX-CUT

Input: a digraph D and an integer k.

Question: is there a vertex partition of D of directed order at least k?

We will show that both D-HAMILTON-PATH and D-MAX-CUT are in FPT when parameterized by directed branch-width. We will do so by first showing some general results providing sufficient conditions which guarantee that a problem is in FPT parameterized by directed branch-width.

One notion important for our sufficient conditions will be that of invariance under source-sink-identifications.

Definition 2.5.4. We call a decision problem Π *source-sink-invariant* if, given any two digraphs D and H equivalent under source-sink-identifications, D is a yes-instance for Π if and only if H also is.

Note that, while problems such as Directed Feedback Vertex Set, Acyclic Coloring and D-MAX-CUT are source-sink-invariant (see the following lemma for D-MAX-CUT), the D-HAMILTON-PATH problem is not. To see this, consider the digraph $J = (\{a, b, c\}, \{\overrightarrow{ab}, \overrightarrow{cb}\})$. Clearly J has no directed Hamiltonian path; however, after performing all possible source-sink-identifications, it does.

Lemma 2.5.5. D-MAX-CUT *is source-sink-invariant.*

Proof. We will first show that, in any instance, there is a vertex partition of maximum order with all sources on the left-hand-side and all sinks on the right-hand-side. Then we will show that identifying the sources or the sinks in any such partition does not change its order.

Take any vertex partition $(V(D) \setminus X, X)$ of any digraph D. Recall that the order of $(V(D) \setminus X, X)$ is the number of edges in the edge separator $S^E_{(V(D)\setminus X, X)}$ consisting of all edges starting in $V(D) \setminus X$ and ending in X. Thus, if there is a source s in X, then no edge incident with s will be in $S^E_{(V(D)\setminus X, X)}$. Hence, letting $Y := X \setminus \{s\}$, the order of $(V(D) \setminus Y, Y)$ is at least that of $(V(D) \setminus X, X)$. Furthermore, a symmetric argument shows that moving all sinks to the right-hand-side of a vertex partition can only increase the order of a partition. Thus we

have shown that, in every digraph D, there exists a vertex partition $(V(D) \setminus Z, Z)$ of maximum order with all sources in $V(D) \setminus Z$ and all sinks in Z.

To conclude the proof, we consider what happens when we identify two sources in D (the proof for identifications of sinks is symmetric). Let s_1 and s_2 be two sources in $V(D) \setminus Z$ and, for $i \in [2]$, let F_i be the set of all edges starting from s_i and ending in Z. Now consider the digraph D' obtained from D by identifying s_1 and s_2 into a new source γ. Since the set of edges starting from γ and ending in Z is precisely $F_1 \cup F_2$, it follows that $|S^E_{(V(D')\setminus Z,Z)}| = |S^E_{(V(D)\setminus Z,Z)}|$. ■

We now show that if a source-sink-invariant problem is in FPT parameterized by underlying tree-width, then it also is when parameterized by directed branch-width.

Theorem 2.5.6. *Let Π be a problem which is in* FPT *parameterized by underlying tree-width and let this fact be witnessed by an algorithm A which, for a computable function f and constant c, decides Π in time at most $f(\mathbf{tw}(u(D'))) \cdot |V(D')|^c$ for any digraph D'. If Π is source-sink-invariant, then Π is in* FPT *parameterized by directed branch-width. Furthermore, there is an algorithm that decides Π in time at most $f(3\,\mathbf{dbw}(D')/2))|V(D')|^c$.*

Proof. First obtain the source-sink-split D of the input digraph D' and note that this can be done in polynomial time (recall that $|V(D)|$ is $O(|V(D')|)$). Since $\mathbf{bw}(u(D)) = \mathbf{dbw}(D')$ (by Lemma 2.4.10) we know that $\mathbf{tw}(u(D)) \leq 3\,\mathbf{dbw}(D')/2$ (by Theorem 2.2.4). Since Π is source-sink-invariant, D will be a yes-instance if and only if D' also is. Thus the result follows since we can apply the FPT algorithm A to solve Π on D. ■

Theorem 2.5.6 allows us to deduce a algorithmic meta-theorem parameterized by directed branch-width by leveraging Courcelle's Meta-theorem for tree-width (Theorem 1.2.7). In particular Theorems 2.5.6 and 1.2.7 imply that the model-checking problem for the subset of source-sink-invariant formulae in the $\mathbf{MSO}_2$-logic of *graphs* is in FPT parameterized by directed branch-width.

Corollary 2.5.7. *Let ϕ be a formula in the $\mathbf{MSO}_2$ logic of graphs. If, for any digraph D' equivalent to D under source-sink identification, we have that $D' \vDash \phi$ if and only if $D \vDash \phi$, then there is an* FPT *algorithm parameterized by $\mathbf{dbw}(D) + |\phi|$ which decides whether D models ϕ.*

We note that Corollary 2.5.7 does not immediately imply the existence of an FPT algorithm for D-HAMILTON-PATH since this is not a source-sink-invariant problem. However, notice that this problem is tractable if the input digraph has more than one source or more than one sink (such a digraph cannot be Hamiltonian).

More generally, for problems that are tractable on inputs with more than some constant number of sources or sinks, we know that either the problem is tractable or we know that the underlying tree-width is bounded in terms of the directed branch-width. Following this train of thought, the following result shows that, problems (such as D-HAMILTON-PATH) which are polynomial-time solvable on digraphs with more than γ sources or sinks (for some fixed γ) and which are also in FPT parameterized by underlying tree-width, belong to FPT when parameterized by directed branch-width.

Corollary 2.5.8. *Let Π be a decision problem on digraphs which is in* FPT *parameterized by underlying tree-width. If there exists a polynomial p and integer γ such that Π can be decided in time $p(|D|)$ on any digraph with at least γ sources or at least γ sinks, then Π is in* FPT *parameterized by directed branch-width.*

Proof. Let D be a digraph and let S and T be the sets of all sources and sinks of D. Check (in time $O(|E(D)|)$) whether $|S| \geq \gamma$ or $|T| \geq \gamma$. If this is the case, then we decide Π in time $p(|D|)$. Otherwise, by Corollary 2.4.8, we know that $\mathbf{bw}(u(D)) \leq \mathbf{dbw}(D) + |S| + |T| \leq \mathbf{dbw}(D) + 2\gamma$. Thus, since the Π is in FPT parameterized by underlying tree-width and since $\mathbf{tw}(u(D)) \leq 3(\mathbf{dbw}(D) + 2\gamma)/2 - 1$ (by Theorem 2.2.4), the result follows. ■

Finally we turn our attention to D-HAMILTON-PATH and D-MAX-CUT. Recall that, when parameterized by underlying tree-width, these problems are in FPT.

Theorem 2.5.9. *[29] For any digraph D,* D-HAMILTON-PATH *and* D-MAX-CUT *can be solved by algorithms running in time at most $f(\mathbf{tw}(u(D)))|D|$ for a computable function f.*

Thus, by Lemma 2.5.5, Theorem 2.5.6 and Corollary 2.5.8 we can the following result that avoids the tower of exponentials given by an application of Courcelle's Theorem.

Corollary 2.5.10. *For any digraph D with directed branch-width k,* D-HAMILTON-PATH *and* D-MAX-CUT *can be solved by algorithms running in time at most $f(k)|D|^2$ for a computable function f.*

Proof. D-MAX-CUT is source-sink-invariant by Lemma 2.5.5. Thus we can solve it in time linear in the number of vertices of the source-sink-split of any input digraph D by Theorems 2.5.9 and 2.5.6. So the result follows for D-MAX-CUT since the source-sink-split has $O(|V(D)|^2)$ vertices. Now notice that any digraph with more than one source or more than one sink is a no-instance of D-HAMILTON-PATH. Thus, by Theorem 2.5.9 and Corollary 2.5.8, the result follows for D-HAMILTON-PATH as well. ■

Since directed branch-width is not closed under directed topological minors, Theorem 2.2.10 does not rule out the existence of a stronger algorithmic meta-theorem. Thus it is natural to ask whether it is possible

to strengthen Corollary 2.5.7 to show fixed-parameter tractability of the $\mathbf{MSO}_2$-model-checking problem (i.e. not only for those formulae which are invariant under source-sink identification) parameterized by directed branch-width. We shall show that this is not possible by demonstrating a problem which is $\mathbf{MSO}_2$-expressible but which is $\mathbf{W}[\mathbf{1}]$-hard when parameterized by directed branch-width. For this purpose, we define the *Directed 2-Reachability Edge Deletion problem* (denoted DRED^2). The 2-*reach of a vertex x in a digraph D* is the set $\text{REACH}^2(D,x)$ of all vertices which are reachable from x in D via a directed path with 0, 1, or 2 edges.

DRED^2.

Input: a digraph D and two naturals k and h and a vertex s of D.

Question: is there an edge-subset F of D of cardinality at most k such that

$$|V(D) \setminus \text{REACH}^2(D-F,s)| \geq h?$$

Lemma 2.5.11. *Every instance (D,k,h,s) of the DRED^2 problem is $\mathbf{MSO}_2$-expressible with a formula of length bounded by a function of k and h.*

Proof. We provide an $\mathbf{MSO}_2$ encoding of the DRED^2 problem via a formula whose length depends only on k and h. Let (D,k,h,s) be an instance of the DRED^2 problem. A digraph D is encoded as a relational structure with ground set $V(D) \cup E(D)$ and a binary incidence relation. We shall write $E(x,y)$ as shorthand to denote that there is an edge $\overrightarrow{xy}$ incident with x and y. For an edge relational variable F and edge-variable $\overrightarrow{xy}$ we write $F(\overrightarrow{xy})$ to denote that $\overrightarrow{xy}$ is in the edge-set F.

Note that there is a directed walk from s to some vertex t with 0, 1 or 2 edges none of which intersect a given edge-set F if and only if at least one of the following formulae holds:

$$\texttt{path}_0(F,t) := s = t \text{ or}$$

$$\texttt{path}_1(F,t) := E(s,t) \wedge \neg F(\overrightarrow{st}) \text{ or}$$

$$\texttt{path}_2(F,t) := \exists x \; E(s,x) \wedge E(x,t) \wedge \neg F(\overrightarrow{sx}) \wedge \neg F(\overrightarrow{xt}).$$

Thus, in a digraph D, a vertex t is in $\textsc{Reach}^2(D-F,s)$ if and only if the following formula holds.

$$\texttt{canReach}(F,t) := \bigvee_{i\in\{0,1,2\}} \texttt{path}_i(F,t) \tag{2.1}$$

Using Equation 2.1, we determine whether there are at least h vertices which cannot be reached from s in $D-F$ with a path on 0, 1 or 2 edges; this is encoded as follows:

$$\texttt{unreachable}_{\geq h}(F) := \exists x_1 \dots \exists x_h \bigwedge_{1\leq i<j\leq h} (x_i \neq x_j) \bigwedge_{i\in[h]} \neg\texttt{canReach}(F,x_i).$$

Next we need to establish a formula that determines whether an edge set F contains at least k edges; we do so as follows:

$$\texttt{card}_{\geq k}(F) := \exists e_1 \dots \exists e_k \bigwedge_{1\leq i<j\leq k} (e_i \neq e_j) \bigwedge_{1\leq i\leq k} F(e_i).$$

Finally we can use the formulae $\texttt{card}_{\geq k}(F)$ and $\texttt{reach}_{\geq h}(F,s)$ to encode the DRED^2 problem as

$$\exists F\ \texttt{unreachable}_{\geq h}(F) \wedge \neg\texttt{card}_{\geq k+1}(F).$$

Note that the length of $\texttt{unreachable}_h(F)$ is quadratically bounded by a function of h and that the length of $\texttt{card}_{\geq k+1}(F)$) is quadratically bounded by a function of k. Thus the formula-length of the entire encoding depends quadratically only on k and h, as desired. ■

We will show that DRED^2 is $\mathbf{W}[\mathbf{1}]$-hard on graphs of bounded directed branch-width when parameterized by h and k. Our proof technique is almost identical to that used by Enright, Meeks, Mertzios, and Zamaraev for an analogous problem in a different setting (temporal graphs) [36]. For completeness, we give full details here.

Theorem 2.5.12 ([36]). *On digraph classes of directed branch-width at most one the* DRED^2 *problem is* $\mathbf{W}[\mathbf{1}]$*-hard when parameterized by the number k of edges which can be deleted and the number h of non-reached vertices after the edge-deletion.*

Proof. We will describe a parameterized reduction from the r-CLIQUE problem (shown to be $\mathbf{W}[\mathbf{1}]$-hard in [35]) which asks whether a given undirected graph G contains an r-clique, where r is taken as the parameter. Let (G,r) be an r-CLIQUE-instance, let $n := |V(G)|$ and $m := |E(G)|$. From (G,r) construct a DRED^2-instance

(D, r, h, s) by defining:

$$V(D) := \{s\} \cup V(G) \cup E(G) \quad \text{(where } s \text{ is a new vertex not in } G\text{)},$$

$$E(D) := \{\overrightarrow{sx} : x \in V(G)\} \cup \{\overrightarrow{xe} : x \in V(G) \text{ and } e \in E(G) \text{ s.t. } x \in e\},$$

$$h := r + \binom{r}{2}.$$

To see that this is indeed a parameterized reduction, note that r clearly bounds itself, h is a function of r and $|V(D)|$ is polynomial in $|V(G)|$. Finally we claim that $\mathbf{dbw}(D) \leq 1$. To see this, take a source-sink-split D' of D and note that $u(D')$ is a collection of disjoint stars centered at elements of $V(G) \cap V(D)$; thus, by Lemma 2.4.10 we have $\mathbf{dbw}(D) = \mathbf{bw}(u(D')) \leq 1$. It remains to be shown that (G, r) is a yes-instance for r-Clique if and only if (D, r, h, s) is a yes-instance for DRED^2.

Note that we can assume $r \geq 3$ since otherwise r-Clique is trivial. Similarly, we may assume that $m \geq r + \binom{r}{2}$ since otherwise there are at most $r+3$ vertices of degree at least $r-1$ and hence we could solve r-Clique on G in time $O(r^3)$.

If (G, r) is a yes-instance, then let U be a vertex-set inducing an r-clique in G. Then the edge-set $F = \{\overrightarrow{su} : u \in U\}$ witnesses that (D, r, h) is a yes-instance since

$$|V(D) \setminus \text{Reach}^2(D - F, s)| = (|U| + |E(G[U])|) \text{ (by the definition of } F\text{)}$$
$$= r + \binom{r}{2} = h \text{ (since } U \text{ induces an } r\text{-clique)}.$$

Conversely, suppose (D, r, h, s) is a yes-instance witnessed by a set F of at most r edges in D such that $|V(D) \setminus \text{Reach}^2(D - F, s)| \geq h$. Let U be the subset of vertices in $V(G) \cap V(D)$ which are incident with an edge in F. Note that $U \subseteq V(G)$ and $|U| \leq r$ by the cardinality of F. Since (D, r, h) is a yes-instance we know that

$$|V(D) \setminus \text{Reach}^2(D - F, s)| \geq (|U| + |E(G[U])|) \text{ (by the definition of } F\text{)}$$
$$\geq r + \binom{r}{2} \text{ (since } (D, r, h, s) \text{ is a yes-instance)}$$

and hence that $|U| + |E(G[U])| \geq r + \binom{r}{2}$. Thus U must induce an r-clique in G. ■

Lemma 2.5.11 and Theorem 2.5.12 show that that, unless FPT = **W**[**1**], the $\mathbf{MSO}_2$-model-checking problem is not in FPT on classes of bounded directed branch-width parameterized by formula length.

2.6 Conclusion and open problems.

We generalized a characterization of classes of bounded tree-width as classes of graphs whose line-graphs have bounded rank-width to the setting of digraphs. We did so by introducing a new digraph width measure called *directed branch-width* and by proving that the classes of digraphs of bounded directed branch-width are exactly those classes whose directed line-graphs have bounded bi-cut-rank-width.

From an algorithmic perspective, we showed that the directed analogues of the Hamilton Path, and Max-Cut problems are in FPT parameterized by directed branch-width. These results are of particular importance since the directed Hamilton Path and Max-Cut problems are intractable when parameterized by any tree-width or rank-width-inspired measure [45, 69]. Furthermore we showed that the model-checking problem for a specific subset of $\mathbf{MSO}_2$-formulae is also in FPT parameterized by directed branch-width.

Directed branch-width opens new doors for the study of digraph connectivity which we describe now. This is partly due to a connection with *tangles* which are objects allowing one to investigate global connectivity properties of graphs (see [87] for a definition). It is known that layouts of a *symmetric sub-modular* function f over a set S are always dual to f-tangles over S [4, 32, 33, 87]. In particular, it is possible to define a notion of *directed tangle* which is dual to directed branch-width.

Tangles and *k-blocks* (introduced in [20]) have been recently abstracted by Diestel, Hundertmark and Lemanczyk [31] via the notion of a *separation profile*. This notion can be applied to graphs, matroids and to data sets. An interesting open problem is that of developing truly directed analogues of k-blocks and separation profiles. The fact that directed branch-width can be associated to a directed tangle makes it a promising starting point for this problem.

To conclude, we point out that Giannopoulou, Kawarabayashi, Kreutzer and Kwon [49] also introduced a notion under the name of 'directed tangle'. However, the two notions are completely incomparable: while our definition of 'directed tangle' is defined with respect to a connectivity function that is symmetric and submodular, the connectivity function that they use is not[2]. It is thus an interesting avenue for further research to compare our notion of 'directed tangle' to theirs.

[2] in their notation, a *directed separation* in a digraph D is a pair (L, R) of *vertex subsets* of D such that $L \cup R = V(D)$ and such that there is no *pair* of edges $\overrightarrow{\ell_1 r_1}$ and $\overrightarrow{r_2 \ell_2}$ which 'cross (L, R) in opposite directions' in D (i.e. s.t. $\{\ell_1, \ell_2\} \subseteq L$ and $\{r, r_2\} \subseteq R$)

3 | Spined categories: generalizing tree-width beyond graphs

3.1 Introduction

Our efforts in Chapter 2 fit in the broader context of finding algorithmically useful notions of 'recursive decomposition' for objects other than finite simple graphs. In this chapter we propose a category-theoretic formalism that provides a meta-answer to the question of defining tree-width analogues in new settings.

The challenge Most notions of recursive graph decomposition are defined in terms of the internal structure of the decomposed object. For example, clique-width decomposition trees – which are used to define the width-measure known as *clique-width* – use a formal grammar to specify how to construct a given graph from smaller ones by adding edges between specific pairs of vertices. For this reason generalizing a given notion of decomposition to a larger class of objects tends to be an arduous task. In fact, as we mentioned in Chapter 2 (and as Kreutzer and Kwon highlight in their survey [68]) even just for digraphs - which are arguably the most similar objects to undirected graphs - the search for an algorithmically useful analogue of tree-width has been an elusive quest that has captivated the research community for the past thirty years. For example, recall from Chapter 2 the myriad of subtly different tree-width analogues that arise when generalizing tree decompositions from simple graphs to digraphs [13, 59, 63, 68, 91].

It is thus clear that, if we wish to find notions of 'recursive decomposability' for more diverse classes of objects, then we need radically new ideas.

We take these considerations to heart in this chapter: we will introduce category-theoretic formalizations

of the notions of 'recursive decomposability', 'recursive decomposition' and 'width measure' which have so far been vague, umbrella terms used to describe the many constructions stemming from the very active field of 'graph width-measures' [26, 29, 30, 52, 82, 93].

Why category theory It is natural to ask why we should we choose to work in the language of category theory. We will settle for a very informal answer for now, but we reassure the reader that this answer will become more concrete already by the end of Section 3.2.

It might be considered a category theorist's motto that 'it is not the *objects* that matter, but the *arrows*'. This is to say that definitions, theorems and proofs in category theory are stated in terms of the *'relationships'* (or 'mappings' or 'homomorphisms') between objects rather than in terms of the objects themselves. Although this might seem like an unnecessary complication at first, this change of perspective has lead to advances and the discovery of deep connections in Mathematics and Computer Science [5, 46, 84]. For example, consider the applications to topology and abstract algebra originating from Eilenberg and Mac Lane's work in the 1940s, or the applications in algebraic geometry stemming from Grothendieck's work in the 1950s, or the categorification of logic stemming from Lawvere's work in the 1960s (just to name a few).

Furthermore, since category-theoretic definitions do not require us to 'see inside' the objects, they are very general and can be applied in a great variety of settings. In fact this will constitute a great strength of our work in this Chapter: we will introduce an abstract definition of tree-decomposition that is 'object agnostic' in the sense that it is defined in terms of the *relations* (i.e. morphisms) between the objects we wish to decompose (i.e. the 'ambient category') rather than in terms of the objects that it decomposes. In particular this means that our work in this chapter opens the doors to finding tree-width analogues for classes of objects that are completely different from graphs (they might come, for example, from algebra or topology) and which do not have obvious counterparts to notions such as vertex, edge, cycle or connectivity.

To accommodate readers from different backgrounds, in Section 3.2 we will give a brief introduction to some standard notions in category theory (such as *functors*, *monomorphisms* and *pushouts*) which we will require in the rest of this Chapter. However, before we do so, we shall give some more graph-theoretical background in Section 3.1.1 which will further contextualise the work in this chapter.

3.1.1 Background and high-level overview

As we already mentioned, in this chapter we will show that the difficulty of transferring a given decomposition notion to a more general setting can be reduced significantly by adopting a category-theoretic perspective. For our purposes, this will amount to finding a characteristic property of tree-width formulated purely in category-theoretic terms and independently of any graph-theoretic notions. Such category-theoretic characterizations have already proven successful in other fields (see e.g. Leinster's work on categorial characterizations of ultraproducts [70] and more recently Lebesgue integration [71]).

Towards finding a category-theoretic formalization of tree-width, however, it is sensible to first seek inspiration from a known definition of tree-width which has a more algebraic flavor compared to – for example – its typical definition in terms of tree decompositions (Definitions 1.2.3 and 1.2.4).

We find such a definition in a paper by Halin from 1976 [55]. There he obtains a characterization of tree-width as a maximal element in a point-wise ordered lattice of functions called *S-functions*. Throughout this chapter we will gradually build-up a theory of categories having sufficient structure for us to prove (Theorem 3.4.15) a vast generalization Halin's characterization.

Before delving into the formal definition of Halin's S-functions, recall (from Definition 1.1.2) that an *H-sum* of two graphs G_1 and G_2 which share a common subgraph H is defined to be a graph $G_1 \#_H G_2$ obtained by identifying an isomorphic copy of H in G_1 with an isomorphic copy of H in G_2.

Observe – as Halin did [55] – that the *Hadwiger number* (the maximum n such that a given graph has a K_n-minor) satsifies the following properties:

1. n-vertex cliques have Hadwiger number n;
2. if we are given a graph G obtained as a clique sum $G \cong G_1 \#_{K_n} G_2$, then the Hadwiger number $h(G)$ of G is equal to its maximum over the addends in the clique-sum (i.e. $h(G) = \max\{h(G_1), h(G_2)\}$);
3. if we are given a graph G which is a minor of a graph H, then the Hadwiger number of G is at most the Hadwiger number of H.

Halin noticed that many other graph invariants (such as the modified chromatic number[1] and the modified connectivity number[2]) behaved similarly (in the above sense) to the Hadwiger number and this prompted him to

[1] The maximum of the chromatic numbers over all minors [55]
[2] One plus the maximum of the connectivity number over all minors [55]

give an axiomatic definition of all functions of this kind. To that end, he defined *S-functions*: these are mappings from finite graphs to the natural numbers satisfying the following axioms.

Definition 3.1.1 ([55]). A function $f : \mathcal{G} \to \mathbb{N}$ is called an *S-function* if it satisfies the following four properties:

(H1) $f(K_0) = 0$ (K_0 is the empty graph)

(H2) if G is a minor of H, then $f(G) \leq f(H)$ (minor isotonicity)

(H3) $f(G \star v) = 1 + f(G)$ (distributivity over adding an apex vertex[a].)

(H4) for each $n \in \mathbb{N}$, $G = G_1 \#_{K_n} G_2$ implies that $f(G) = \max_{i \in \{1,2\}} f(G_i)$ (distributivity over clique-sum).

[a] Recall from Chapter 1 that a vertex v is an apex if it is adjacent to every other vertx in the graph

The set of all S-functions can be seen to form a poset under the pointwise ordering (i.e. for S-functions f and g, we write $f \leq g$ if $f(G) \leq g(G)\ \forall G \in \mathcal{G}$); in fact it is a very large poset since Halin showed that there are uncountably many S-funtions [55]. While studying the order-theoretic properties of S-functions, he obtained the following characterization of tree-width as *the maximal S-function*.

Theorem 3.1.2 ([55]). *The set of all S-functions forms a complete distributive lattice when equipped with the pointwise ordering. Furthermore, the function $G \mapsto \mathbf{tw}(G) + 1$ is maximal in this lattice.*

Before moving on, we take a moment to address the question of what is gained by seeking a generalization of Halin's Theorem 3.1.2.

The brief answer is that understanding S-functions tells us much more than simply understanding tree-width since S-functions describe a large class of graph invariants (including ones such as the modified chromatic number and the Hadwiger number) that have played central role in many aspects of graph theory and complexity theory.

For a more concrete answer to why we should seek a generalization of Theorem 3.1.2, first notice that the pointwise ordering on S-functions can be equivalently stated as an ordering on *graph classes*. To see this, take any two S-functions f and g with $f \leq g$ and observe that any graph G in the class $g_{\leq n} := \{G \in \mathcal{G} : g(G) \leq n\}$ must also be an element of the class $f_{\leq n}$ since $f(G) \leq g(G) \leq n$ (because we assumed that $f \leq g$ in the point-wise order). In particular this means that $f \leq g$ if and only if $f_{\leq n} \supseteq g_{\leq n}$ for all $n \in \mathbb{N}$. In the context of Theorem 3.1.2, these observations imply that a graph class $\mathbf{K}$ has tree-width at most k if and only if, letting Ξ denote the set

of all S-functions, we have $\mathbf{K} \subseteq \bigcap_{f \in \Xi} f_{\leq k}$. These observations thus imply that both Halin's Theorem 3.1.2 and our generalization thereof (i.e. Theorem 3.4.15) provide us with a very deep understanding of classes of bounded tree-width: if we wish to show that a graph-class has *unbounded* tree-width, all we need to do is find *one* S-function which is unbounded on that class. This is a powerful observation since there are S-functions that have a more local nature and which are significantly easier to study compared to tree-width [55].

As we mentioned, our main theorem in this chapter (Theorem 3.4.15) will be a vast generalization of Theorem 3.1.2 which will result in our abstract characterization of tree-width. To obtain this generalization, we introduce *spined categories* which provide a precise, abstract definition of recursive decomposability. Roughly, spined categories are categories equipped with sufficient additional structure to admit both

- a well-behaved notion of recursive decomposition (which we call *pseudo-chordal completions*) and
- a categorial generalization of the graph-theoretic notion of tree-width (we call this the *triangulation functor* and we construct it in Section 3.4).

Spined categories come equipped with a *proxy-pushout* operation whose role is largely analogous to that of the clique-sum operation in Halin's definition of S-functions (Definition 3.1.1). The role of cliques themselves is played by the members of a distinguished sequence of objects, called the *spine*.

Among the structure-preserving mappings (i.e. functors) between spined categories, we find abstract (and functorial) counterparts to Halin's S-functions: we call these *S-functors* (Definition 3.3.7). We shall see that S-functors are in fact more general than Halin's notion, even in the case of simple graphs. While every S-function yields an S-functor over the category of graphs and injective homomorphisms (Proposition 3.3.8), the converse is not true.

We show that, when appropriately instantiated, the triangulation functors (i.e. our abstract analogues of tree-width) encompass several tree-width-like invariants such as hypergraph tree-width (Theorem 3.4.17), and the tree-width of the modular quotient in the category of modular partition functions (Example 3.5.2).

Our uniform construction of triangulation functors allows one to define new tree-width-like parameters (such as widths for new types of combinatorial objects, or notions of graph width-measures that respect different notions of structural correspondence than ordinary graph isomorphism) simply by collecting the relevant objects into a spined category. As such, we shall see that the results in this chapter can bee seen as providing a 'black-box' taking as input a spined category (i.e. some class of objects we wish to decompose) and yielding as output an

appropriate analogue of tree-width.

The rest of this chapter is structured as follows. First we will give a gentle introduction to all of the needed category-theoretic notions in Section 3.2. We will then introduce our new notion of a *spined category* and investigate its basic properties in Section 3.3. Section 3.4 contains the proof of our main result (Theorem 3.4.13) on the existence of spined functors called *triangulation functors* which encompass and generalize several tree-width-like invariants used in combinatorics. In Section 3.5 we describe a way of constructing new spined categories from previously known ones and illustrate the applications of such constructions with some examples. Section 3.6 briefly discusses open questions and directions for future research.

3.2 Category-theoretic preliminaries

Throughout this section we will follow the notation found in Awodey's textbook [5]. Two other introductory textbooks that might be more to the reader's taste (depending on backgrounds) are Riehl's [84] or Fong and Spivak's [46] textbooks (the first is focused on 'pure' category theory and draws many examples from topology; the second focused on applications of category theory).

Definition 3.2.1. A *category* C consists of:

- a collection of *objects*: $A, B, C \ldots$ denoted $\mathbf{Ob}(C)$ and
- a collection of *arrows* (or *morphisms*): $f, g, h \ldots$ denoted $\mathbf{Mor}(C)$,

such that:

- every arrow f is associated to two (not necessarily distinct) objects $\mathbf{dom}(f)$ and $\mathbf{cod}(f)$ called the *domain* and *codomain* of f (we write $f : A \to B$ whenever $A = \mathbf{dom}(f)$ and $B = \mathbf{cod}(f)$)
- for all pairs of arrows f and g with $\mathbf{cod}(f) = \mathbf{dom}(g)$, there is an arrow $g \circ f : \mathbf{dom}(f) \to \mathbf{cod}(g)$ called the *composite* of f with g (or simply 'g *after* f'),
- for each object A there is an arrow $\mathbb{1}_A : A \to A$ called the *identity arrow of* A

and subject to the following two laws:

- associativity of composition: for any three arrows $A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D$ we have $h \circ (g \circ f) = (h \circ g) \circ f$.
- identities are 'units' under both the pre- and post-composition operations: for all arrows $f : A \to B$, we have $f \circ \mathbb{1}_A = f = \mathbb{1}_B \circ f$.

For any two objects A and B in a category $\mathcal{C}$, we call the collection of all arrows from A to B (i.e. all arrows f with $\mathbf{dom}(f) = A$ and $\mathbf{cod}(f) = B$) the *hom-set* from A to B and we denote it as $\mathbf{Hom}(A, B)$.

We will call a category $\mathcal{C}$ *small* if both its collection of objects and its collection of arrows are sets; otherwise we say that $\mathcal{C}$ is *large*. Throughout this thesis we will assume all categories to be small unless explicitly stated otherwise. Furthermore, to aid readability, we will often drop the composition symbol '$\circ$' by simply writing gf to denote the composite $g \circ f$.

Examples of categories are everywhere. A familiar category is **FinSet**; it has *finite sets* as objects and *functions* between them as arrows. It is easy to check that every set has an identity arrow and that arrow composition is associative.

Any *poset* is an example of a special kind of category called a *poset category*[3]: these are categories in which, for any two objects A and B, there is *at most one* arrow from A to B. In particular this means that the usual ordering $\leq$ on the natural numbers turns the partially-ordered set $\mathbb{N}_{\leq}$ into a category in which there is an arrow $n \to m$ if and only if $n \leq m$. The same is true for the poset $\mathbb{N}_{=}$ given by the natural numbers under the equality relation (i.e. there is an arrow $n \to m$ if and only if $n = m$).

Two other examples of categories are $\mathbf{Gr}_{homo}$ and $\mathbf{HGr}_{homo}$. The first is the category of *graphs* and *graph homomorphisms* (i.e. each arrow $f : G \to H$ is a graph homomorphism from G to H). The second is the category of *hypergraphs* and *hypergraph homomorphisms*. We point out that these categories differ from what is known as 'the category of graphs' in category theory [5]: we will *not* assume our graphs and hypergraphs to be reflexive[4].

Many more examples of categories are found in familiar mathematical objects and their respective structure preserving mappings [5, 46, 84]; these include:

- groups and group homomorphisms,

[3] The converse is not true: although every poset category is easily seen to give rise to a *pre-order* (sometimes called a quasi-order), there is no guarantee in general that this will also be a partial order.

[4] recall that G is reflexive there is a loop-edge from every vertex to itself in G.

- topological spaces and continuous mappings, and
- posets and order-preserving maps.

Since all posets are categories (as we mentioned earlier) this last example shows that we can have categories whose objects are themselves categories. This can be defined more generally via an appropriate notion of 'structure-preserving mappings' between categories: these are called *functors*. Although we will not make use of it here, we point out that this notion can be used to define a large category **Cat** which has *small categories as objects* and *functors as arrows*.

Definition 3.2.2. A *functor* F from a category $\mathcal{C}$ to a category $\mathcal{D}$ is a mapping that associates

- to every object W in $\mathcal{C}$ an object $F[W]$ in $\mathcal{D}$
- to every arrow $f : X \to Y$ in $\mathcal{C}$ an arrow $F[f] : F[X] \to F[Y]$ in $\mathcal{D}$

in such a way that identities and compositions are preserved; formally this amounts to the following requirements:

- $F[\mathbb{1}_X] = \mathbb{1}_{F[X]}$ for every object X in $\mathcal{C}$, and
- $F[g \circ f] = F[g] \circ F[f]$ for all arrows $f : X \to Y$, $g : Y \to Z$ in $\mathcal{C}$.

Notice that any functor $F : \mathcal{C} \to \mathcal{D}$ induces mappings

$$F_0 : \mathbf{Ob}(\mathcal{C}) \to \mathbf{Ob}(\mathcal{D}),$$

$$F_1 : \mathbf{Mor}(\mathcal{C}) \to \mathbf{Mor}(\mathcal{D}) \text{ and}$$

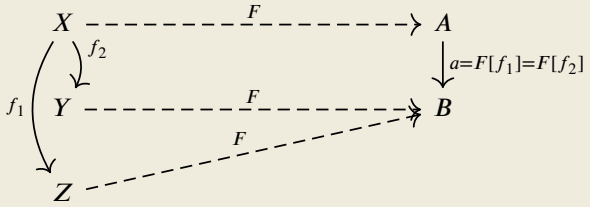
$$F_{X,Y} : \mathbf{Hom}_{\mathcal{C}}(X,Y) \to \mathbf{Hom}_{\mathcal{D}}(F[X], F[Y]) \qquad \text{(where } (X,Y) \text{ is any pair of objects in } \mathcal{C}\text{)}.$$

Depending on whether F_0, F_1 and $F_{X,Y}$ are injective or surjective, we have different notions of 'injectivity' and 'surjectivity' that are definable for F; thus we call F:

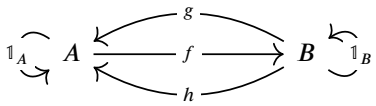
- *injective* (resp. *surjective*) on *objects* if F_0 is injective (resp. *surjective*),
- *injective* (resp. *surjective*) on *arrows* if F_1 is injective (resp. *surjective*) and
- *faithful* (resp. *full*) if $F_{X,Y}$ is injective (resp. *surjective*) for all pairs (X,Y) of objects in $\mathcal{C}$.

Example 3.2.3. *A functor that is injective on arrows is also faithful; however the converse is not true. For example, consider the functor $F : \mathcal{C} \to \mathcal{D}$ depicted (dashed) below (identities are omitted). It is faithful, but neither injective on arrows nor injective on objects.*

The category $\mathcal{C}$ *The category $\mathcal{D}$*



We say that an arrow $f : A \to B$ in any category $\mathcal{C}$ is an *isomorphism* if there exists an arrow $f^{-1} : B \to A$ called '*the inverse of f*' such that $f f^{-1} = \mathbb{1}_A$ and $f^{-1} f = \mathbb{1}_B$. To see that isomorphisms are unique, consider any diagram of the following form.



If $fh = \mathbb{1}_B$ and $gf = \mathbb{1}_A$, then $g = g\,\mathbb{1}_B = gfh = \mathbb{1}_A h = h$. In particular, since identities are their own inverses, they are isomorphisms and hence they are unique.

Note that, although categorical isomorphisms correspond to bijective functions in **FinSet**, this relationship to bijections is not true in general: in the category $\mathbf{Gr}_{homo}$ of graphs and graph-homomorphisms, there can be set-theoretic bijections between two non-isomorphic graphs (in contrast, the category-theoretic notion of isomorphism does indeed coincide with its graph-theoretic counterpart) [5].

Two important notions from set theory are injective and surjective functions. Both of these notions have category-theoretic generalizations via the notions of *mono-* and *epi-morphisms*. We note that, as was the case with isomorphisms, although monomorphisms and epimorphisms do indeed coincide respectively with injections and surjections in **FinSet**, this is not true in arbitrary categories [5].

Definition 3.2.4. We call an arrow $f : A \to B$ in a category $\mathcal{C}$ a *monomorphism* in $\mathcal{C}$ (or a *monic* arrow in $\mathcal{C}$) if, given any two arrows $x, y : Z \to A$, we have $f \circ x = f \circ y$ implies $x = y$. Similarly, f is an *epimorphism* (or an *epic* arrow) if, given any two arrows $x, y : B \to C$, we have that $x \circ f = y \circ f$ implies $x = y$. Throughout this text the notation $f : A \hookrightarrow B$ always denotes a monomorphism from A to B while we denote epimorphisms as $A \twoheadrightarrow B$.

Note that, the definitions of monic and epic arrows are symmetrical in the sense that we can obtain one from the other by simply 'reversing the directions of all arrows' (and changing the order of composition). This is an instance of the fundamental concept of *categorical duality*: given any category $\mathcal{C}$, we denote by $\mathcal{C}^{\mathbf{op}}$ the *dual* or *opposite* category to $\mathcal{C}$ obtained by reversing all arrows in $\mathcal{C}$. For more on categorical duality, we refer the reader to Awodey's textbook [5].

A category $\mathcal{D}$ is a *subcategory* of a category $\mathcal{C}$ if $\mathbf{Ob}(\mathcal{D})$ and $\mathbf{Mor}(\mathcal{D})$ are contained in $\mathbf{Ob}(\mathcal{C})$ and $\mathbf{Mor}(\mathcal{C})$ respectively (note that this gives rise to the obvious functor $\mathcal{D} \to \mathcal{C}$ which embeds $\mathcal{D}$ into $\mathcal{C}$). One subcategory that we highlight here is $\mathbf{Mono}(\mathcal{C})$ which has $\mathbf{Ob}(\mathbf{Mono}(\mathcal{C})) = \mathbf{Ob}(\mathcal{C})$ and which has as arrows only those arrows in $\mathcal{C}$ which are monomorphisms[5].

3.2.1 Universal constructions

The main aspect of 'the category-theoretic-thinking' that we will leverage in this thesis is that of formulating definitions by way of 'universal constructions'. Roughly these are ways of defining concepts (e.g. the product of two sets) by speaking only about the *arrows* in the category (e.g. functions from the product); definitions by way of a universal construction have the benefit that they can be immediately applied to any other category (thus generalizing the original notion that they formalized).

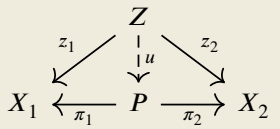
This section will review some simple universal constructions such as *products*, *co-products* and *pushouts*. Note that we will not make direct use of these notions in Chapter 3; however, since knowledge of these notions will aid in the understanding the constructions in Chapter 3, we take the time to review these concepts here.

We call any pair of arrows $g : A \to G$ and $h : A \to H$ with the same domain in the category $\mathcal{C}$ a *span in* $\mathcal{C}$ and we call the dual diagram $G \xrightarrow{g} A \xleftarrow{h} H$ a *cospan*. A *monic* (co)span is a (co)span consisting of monic arrows.

The first universal construction that we shall define is that of a *product* of two objects in a category.

Definition 3.2.5. The *product* of two objects X_1 and X_2 in a category $\mathcal{C}$ is a *span* $X_1 \xleftarrow{\pi_1} P \xrightarrow{\pi_2} X_2$ in $\mathcal{C}$ such that, for any other span of the form $X_1 \xleftarrow{z_1} Z \xrightarrow{z_2} X_2$ there is a *unique* arrow $u : Z \to P$ such that $z_i = \pi_i u$; in other words, the following diagram is required to commute[a]

[5] Note that this notation differs from certain usages, where $\mathbf{Mono}(\mathcal{C})$ instead denotes a specific subcategory of the 'arrow category' of $\mathcal{C}$.



(i.e. $\pi_i u = z_i$ for $i \in [2]$).

[a] A diagram is said to commute, if every two arrows (including composites) which have the same domain and codomain are equal. Formally, by *diagram of shape* $\mathcal{J}$ we mean a functor $D : \mathcal{J} \to \mathcal{C}$; to aid readability (and to avoid keeping devoting notation to functors which are only used to define diagrams) we will not state the functor explicitly and instead simply refer to some drawn collection of arrows (i.e. the directed multi-graph that represents the diagram).

As an example, consider the category **FinSet**: here categorical products coincide with set-theoretic products and the maps π_1 and π_2 correspond to the projection maps from the product of two sets to its factors[6] [5]. In fact, to highlight this correspondence, the *product object* P from Definition 3.2.5 is often denoted as $X_1 \times X_2$.

Notice that, if $\mathcal{C}$ is a partial order, then the product of two objects in $\mathcal{C}$ is simply their *infimum*. In particular one should observe that, since not all posets have all infima (e.g. any poset isomoprhic to an antichain), it follows that *categories need not have products* (i.e. there might be two objects that do not have a product in $\mathcal{C}$).

By duality[7], we also have the notion of a *co-product* of two objects X_1 and X_2 in $\mathcal{C}$. It is easy to see that, whenever $\mathcal{C}$ is a partial order, the co-product of two objects in $\mathcal{C}$ is simply their *supremum*. Another example of an instantiation of the notion of a co-product is found in $\mathbf{Gr}_{homo}$: it is not hard to show that the coproduct of two graphs is given by their disjoint union [5] (in fact this is also true in **FinSet**).

Another universal construction (similar to coproducts) is given by the notion of a *pushout*. We point out that this notion will be of particular use to us in the next Chapter: specifically in our definition of *proxy-pushouts* (Definition 3.3.1).

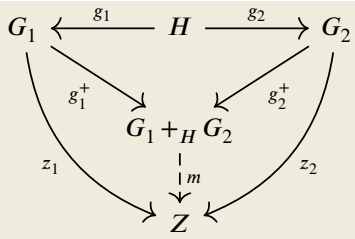
Definition 3.2.6. Consider a span $G_1 \xleftarrow[g_1]{} H \xrightarrow[g_2]{} G_2$ in a category $\mathcal{C}$. We call a cospan of the form $G_1 \xrightarrow{g_1^+} G_1 +_H G_2 \xleftarrow{g_2^+} G_2$ a *pushout of* g_1 *and* g_2 *in* $\mathcal{C}$ if

1. $g_1^+ \circ g_1 = g_2^+ \circ g_2$, and
2. for any cospan $G_1 \xrightarrow{z_1} Z \xleftarrow{z_2} G_2$ such that $z_1 \circ g_1 = z_2 \circ g_2$ there exists a *unique* morphism $m : G_1 +_H G_2 \to Z$ such that $m \circ g_1^+ = z_1$ and $m \circ g_2^+ = z_2$;

equivalently, we require the following diagram to commute.

[6] for example, taking the two sets $S_1 := \{1,2\}$ and $S_2 := \{3\}$, we would have that the projection $\pi_1 : S_1 \times S_2 \to S_1$ maps $(1,3)$ to 1 and $(2,3)$ to 2.

[7] we can simply *define* the co-product of two objects X_1 and X_2 to be their product in $\mathcal{C}^{\mathbf{op}}$



(In summary, we require g_1^+ and g_1^+ to satisfy $g_1^+ g_1 = g_2^+ g_2$ s.t.

$\forall z_1, z_2, \exists! m$ satisfying $z_1 g_1 = z_2 g_2$, s.t. $m g_i^+ = z_i$ for $i \in [2]$.)

We call $G_1 +_H G_2$ the *pushout object of* g_1 *and* g_2.

It is easy to show that, if they exist, products, co-products and pushouts are unique.

From a graph-theoretic point of view, pushouts allow us to generalize the notion of an H-sum of two graphs (i.e. Definition 1.1.2 or, informally: 'glueing together two graphs along a common subgraph H').

Proposition 3.2.7 (Folklore)**.** *Every monic span in* $\mathbf{Gr}_{homo}$ *has a pushout. In particular, the pushout of a monic span* $G_1 \xhookleftarrow{g_1} H \xhookrightarrow{g_2} G_2$ *is the graph given by the following* H*-sum of* G_1 *and* G_2

$$G_1 +_H G_2 := (G_1 \uplus G_2)/_{g_1 = g_2}.$$

Proof. Take the obvious inclusion maps as $\iota_1 : G_1 \hookrightarrow G_1 \#_H G_2$ and $\iota_2 : G_2 \hookrightarrow G_1 \#_H G_2$. We clearly have $\iota_1 \circ g_1 = \iota_2 \circ g_2$. Now consider any other cospan $G_1 \xrightarrow{z_1} Z \xleftarrow{z_2} G_2$ satisfying the equality $z_1 \circ g_1 = z_2 \circ g_2$. Define the map $m : G_1 \#_H G_2 \to Z$ on the vertices of $G_1 \#_H G_2$ via the equation

$$m(v) = \begin{cases} z_1(v) & \text{if } v \text{ is in the range of } \iota_1, \\ z_2(v) & \text{otherwise.} \end{cases}$$

To check that $m \circ \iota_1 = z_1$, it suffices to prove $m(\iota_1(x)) = z_1(x)$ for an arbitrary vertex x of G. Since $\iota_1(x) = x$ and $x \in G$, the first clause of the definition applies, and we have $m(\iota_1(x)) = m(x) = z_1(x)$. A similar proof allows us to conclude $m \circ \iota_2 = z_2$. The uniqueness of m follows immediately. ■

We cannot generalize Proposition 3.2.7 much further since the pushout of an arbitrary span of the form $G \xleftarrow{i} D \xrightarrow{j} H$ need not exist in $\mathbf{Gr}_{homo}$. Indeed, taking the obvious injection $i : \overline{K_2} \to K_2$ and the unique map $j : \overline{K}_2 \to K_1$, we see that no object Z and maps $z_i : K_i \to Z$ (where $i \in [2]$) can ever satisfy $z_1 \circ i = z_2 \circ j$, since the image of the right-hand side always consists of a single vertex, while the image of the left-hand side necessarily contains an edge.

3.3 Introducing spined categories and S-functors

In what follows, notice that any functor $\Omega : \mathbb{N}_{=} \to \mathcal{C}$ (where $\mathbb{N}_{=}$ is the discrete category with an arrow $x \to y$ iff $x = y$) describes a *sequence* $(\Omega[n])_{n\in\mathbb{N}}$ of objects in $\mathcal{C}$. To highlight this fact and to ease readability, we often drop the brackets and write Ω_n rather than $\Omega[n]$.

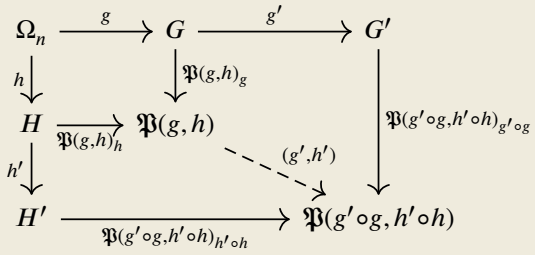
Definition 3.3.1. A *spined category* consists of a category $\mathcal{C}$ equipped with the following additional structure:

- a functor $\Omega : \mathbb{N}_{=} \to \mathcal{C}$ called the *spine* of $\mathcal{C}$,
- an operation $\mathfrak{P}$ (called the *proxy-pushout*) that assigns to each span of the form $G \xleftarrow{g} \Omega_n \xrightarrow{h} H$ in $\mathcal{C}$ a distinguished cospan $G \xrightarrow{\mathfrak{P}(g,h)_g} \mathfrak{P}(g,h) \xleftarrow{\mathfrak{P}(g,h)_h} H$

subject to the following conditions:

SC1 For every object X of $\mathcal{C}$ there exists a morphism $x : X \to \Omega_n$ for some $n \in \mathbb{N}$.

SC2 For any cospan $G \xleftarrow{g} \Omega_n \xrightarrow{h} H$ and any pair of morphisms $g' : G \to G'$ and $h' : H \to H'$ there exists a *unique* morphism $(g', h') : \mathfrak{P}(g,h) \to \mathfrak{P}(g' \circ g, h' \circ h)$ making the following diagram commute:



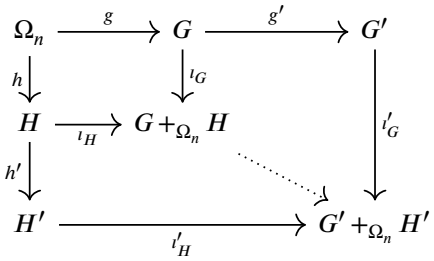
One could define *proxy pullbacks* dually to proxy-pushouts. As the name suggests (and as we shall prove in Proposition 3.3.2), categories that have pushouts for every span of the form $G \xleftarrow{g} \Omega_n \xrightarrow{h} H$ always have proxy-pushouts (dually it is easily seen that categories having pullbacks of all diagrams of the form $G \xrightarrow[g]{} \Omega_n \xleftarrow[h]{} H$ have proxy pullbacks). This observation gives rise to many examples of spined categories.

Proposition 3.3.2. *Take a category $\mathcal{C}$ equipped with functor $\Omega : \mathbb{N}_{=} \to \mathcal{C}$ such that the following hold:*

1. *for any object X of $\mathcal{C}$ there is some $n \in \mathbb{N}$ and morphism $x : X \to \Omega_n$, and*
2. *every span of the form $G \xleftarrow{g} \Omega_n \xrightarrow{h} H$ has a pushout in $\mathcal{C}$.*

The map $\mathfrak{P}$ that assigns to every span $G \xleftarrow{g} \Omega_n \xrightarrow{h} H$ its pushout co-span turns $\mathcal{C}$ into a spined category.

Proof. We only have to verify Property **SC2**. Consider the diagram



We have to exhibit the unique dotted morphism $G +_{\Omega_n} H \to G' +_{\Omega_n} H'$ making this diagram commute. Since $G +_{\Omega_n} H$ is a pushout of g and h and since the arrows $\iota'_G \circ g'$ and $\iota'_H \circ h'$ form a co-span which commutes with the span of g, h[8], the existence and uniqueness of the required morphism $G +_{\Omega_n} H \to G' +_{\Omega_n} H'$ follows (by the definition of a pushout; see Definition 3.2.6). ■

Since pushouts in poset categories are given by least upper bounds, Proposition 3.3.2 allows us to construct a simple (but important) first example of a spined category.

Example 3.3.3. *Let $\leq$ denote the usual ordering on the natural numbers. The poset $\mathbb{N}_{\leq}$, when equipped with the spine $\Omega_n = n$ (and maxima as proxy-pushouts) constitutes a spined category denoted* **Nat**.

Another easy example is given by the category **FinSet**.

Example 3.3.4. *Take the sequence $(\{1, 2, \ldots, n\})_{n \in \mathbb{N}}$ as the spine (observe that it trivially satisfies Property **SC1**). Since* **FinSet** *has all pushouts, by Proposition 3.3.2, we know that it has proxy-pushouts (thus satisfying Property **SC2**).*

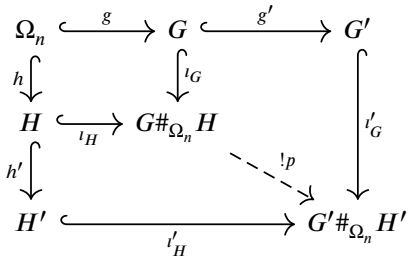
Combining Propositions 3.2.7 and 3.3.2 gives us a first example of a combinatorial spined category, the category $\mathbf{Gr}_{mono}$ which has graphs as objects and *injective* graph homomorphisms as arrows. First consider a span of the form $A \xhookleftarrow{a} X \xhookrightarrow{b} B$ in $\mathbf{Gr}_{homo}$. Notice that all arrows are monic in the corresponding

[8] i.e. the arrows $\iota'_G \circ g'$ and $\iota'_H \circ h'$ form a so-called "co-cone" to g, h.

pushout square. However, given a co-span $A \xrightarrow{a'} Z \xleftarrow[b']{} B$ the pushout morphism $A +_X B \to Z$ can fail to be injective (i.e. a monomorphism) for instance in the case where the images of a' and b' have non-empty intersection. To see this, take $A \cong K_2 \cong B$ and $X \cong K_1$ (which implies that $A\#_X B \cong P_3$). It is easy to see that we can choose $Z \cong K_2$ and appropriate a' and b' satisfying $a'a = b'b$; however the unique pushout arrow $A\#_X B \to K_2$ is clearly not injective. It follows that the clique sum does not give rise to pushouts in the category $\mathbf{Gr}_{mono}$. However, this is not a problem since clique-sums *do* give rise to *proxy* pushouts in $\mathbf{Gr}_{mono}$.

Proposition 3.3.5. *The category $\mathbf{Gr}_{mono}$, equipped with the spine $n \mapsto K_n$ and clique sums as proxy-pushouts forms a spined category.*

Proof. Property **SC1** is evident, but we need to verify Property **SC2**. Consider the diagram



in $\mathbf{Gr}_{homo}$. Notice that the arrows $\iota_G, \iota'_G, \iota_H, \iota'_H$ are all monic. We have to establish that the morphism $p : G\#_{\Omega_n} H \to G'\#_{\Omega_n} H'$ (which is unique since it is a pushout arrow in $\mathbf{Gr}_{homo}$) is monic (this will allow us to conclude that p is an arrow of $\mathbf{Gr}_{mono}$ as well). Note that p maps any vertex x in $G\#_{\Omega_n} H$ to $(\iota'_G \circ g')(x)$ if x is in G (i.e. if it is in the range of ι_G) and to $(\iota'_H \circ h')(x)$ otherwise. Thus, since $V(G') \cap V(H') = V(G) \cap V(H)$, we have that, for any x and y in $V(G\#_{\Omega_n} H)$, if $p(x) = p(y)$ then $x = y$. Thus p is injective (i.e. it is monic and hence it is in $\mathbf{Gr}_{mono}$). ■

We will encounter further examples of spined categories below, including:

1. the poset of natural numbers under the divisibility relation (Proposition 3.3.13),
2. the category of posets (Proposition 3.3.11),
3. the category of hypergraphs (Theorem 3.4.17),
4. the category of vertex-labelings of graphs (Examples 3.5.2 and 3.5.3).

Now we introduce the notion of a *spined functor* as the obvious notion of morphism between two spined categories: to be spined, a functor has to preserve both the spine and proxy-pushouts.

Definition 3.3.6. Consider spined categories $(\mathcal{C}, \Omega^{\mathcal{C}}, \mathfrak{P}^{\mathcal{C}})$ and $(\mathcal{D}, \Omega^{\mathcal{D}}, \mathfrak{P}^{\mathcal{D}})$. We call a functor $F : \mathcal{C} \to \mathcal{D}$ a *spined functor* if it

SF1 *preserves the spine*, i.e. $F \circ \Omega^{\mathcal{C}} = \Omega^{\mathcal{D}}$, and

SF2 *preserves proxy-pushouts*, i.e. given a proxy-pushout square

$$\begin{array}{ccc} \Omega^{\mathcal{C}}_n & \xrightarrow{g} & G \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle \mathfrak{P}^{\mathcal{C}}(g,h)_g} \\ H & \xrightarrow[\mathfrak{P}^{\mathcal{C}}(g,h)_h]{} & \mathfrak{P}^{\mathcal{C}}(g,h) \end{array}$$

in the category $\mathcal{C}$, the image

$$\begin{array}{ccc} \Omega^{\mathcal{D}}_n & \xrightarrow{Fg} & F[G] \\ {\scriptstyle Fh}\downarrow & & \downarrow{\scriptstyle F\mathfrak{P}^{\mathcal{C}}(g,h)_g} \\ F[H] & \xrightarrow[F\mathfrak{P}^{\mathcal{C}}(g,h)_h]{} & F[\mathfrak{P}^{\mathcal{C}}(g,h)] \end{array}$$

is isomorphic to a proxy-pushout square in $\mathcal{D}$. Equivalently, $F[\mathfrak{P}^{\mathcal{C}}(g,h)] \cong \mathfrak{P}^{\mathcal{D}}(Fg, Fh)$, $F\mathfrak{P}^{\mathcal{C}}(g,h)_g \cong \mathfrak{P}^{\mathcal{D}}(Fg, Fh)_{Fg}$ and $F\mathfrak{P}^{\mathcal{C}}(g,h)_h \cong \mathfrak{P}^{\mathcal{D}}(Fg, Fh)_{Fh}$ all hold.

Recall the spined category **Nat** of Example 3.3.3. Using spined functors to **Nat**, we obtain the following categorial counterparts to Halin's S-functions.

Definition 3.3.7. An *S-functor* over the spined category $\mathcal{C}$ is a spined functor $F : \mathcal{C} \to \mathbf{Nat}$.

Proxy pushouts in $\mathbf{Gr}_{mono}$ are given by clique sums over complete graphs, while proxy-pushouts in **Nat** are given by maxima. Consequently, given an S-functor $F : \mathbf{Gr}_{mono} \to \mathbf{Nat}$, Property **SF2** ensures us that evaluating F on a clique-sum (i.e. the proxy-pushout operation in $\mathbf{Gr}_{mono}$) is the same as evaluating F on the addends of the clique-sum and then taking a maximum (i.e. the proxy-pushout operation in **Nat**); in other words we have $F[G \#_{K_n} H] = \max\{F[G], F[H]\}$ (cf. Property **(H4)** of Halin's S-functions). This observation allows us to deduce that every one of Halin's S-functions gives rise to an S-functor over $\mathbf{Gr}_{mono}$.

Proposition 3.3.8. *Every S-function $f : \mathcal{G} \to \mathbb{N}$ gives rise to an S-functor F satisfying $F[X] = f(X)$ for all objects X of $\mathbf{Gr}_{mono}$.*

Proof. Take an S-function $f : \mathcal{G} \to \mathbb{N}$. Take a morphism $x : X \to Y$ in $\mathbf{Gr}_{mono}$. Since x is a graph monomorphism, X is isomorphic to a subgraph of Y, and is therefore a (trivial) minor of Y. Thus, since S-functions are minor isotone (Property **(H2)**), we have that $f(X) \leq f(Y)$ holds. It follows that the map F defined by the equations

$F[X] = f(X)$ and $Ff = (F[X] \leq F[Y])$ for each pair of objects X, Y and each morphism $x : X \to Y$ constitutes a functor from $\mathbf{Gr}_{mono}$ to the poset category $\mathbb{N}_{\leq}$.

We show that F preserves the spine inductively, by proving $F[K_n] = f(K_n) = n$ for all $n \in \mathbb{N}$. For the base case, notice that we have $F[K_0] = 0$ by Property **(H1)**. For the inductive case, assume that $F[K_n] = f(K_n) = n$. Since $K_n \star v = K_{n+1}$, we have $F[K_{n+1}] = f(K_{n+1}) = f(K_n \star v) = 1 + f(K_n) = 1 + n$ by Property **(H3)**.

Finally, as we observed earlier, the preservation of proxy-pushouts follows immediately by Property **(H4)**. Hence F is a spined functor as we claimed. ■

We note, however that the converse of Proposition 3.3.8 does not hold (not even in $\mathbf{Gr}_{mono}$). To see this, consider the clique-number. It is an S-functor in $\mathbf{Gr}_{mono}$, since it cannot increase when taking subgraphs (i.e. it satisfies Property **SF1**) and the clique-number of any clique-sum $G\#_{K_n}H$ it given by the maximum of the clique-numbers of the addends G and H (i.e. $\omega(G\#_{K_n}H) = \max\{\omega(G), \omega(H)\}$ thus satisfying Property **SF2**). In contrast, it is easy to see that the clique-number may increase when taking minors. Thus the lack of minor-isotonicity shows that, despite being an S-functor, the clique number does not satisfy Property **(H2)** and hence it is not an S-function.

Using the natural indexing on the spine given by the functor $\Omega : \mathbb{N}_{=} \to \mathcal{C}$, we can associate the following numerical invariants to each object of the spined category $\mathcal{C}$.

Definition 3.3.9. Take a spined category $(\mathcal{C}, \Omega, \mathfrak{P})$ and an object $X \in \mathcal{C}$. We define the *order* of the object X (written $|X|$) to be the least $n \in \mathbb{N}$ such that $\mathcal{C}$ has a morphism $X \to \Omega_n$. Similarly, we define the *generalized clique number* of the object X (written $\dot{\omega}(X)$) as the largest $n \in \mathbb{N}$ for which $\mathcal{C}$ contains a morphism $\Omega_n \to X$ (whenever such n exists).

It's clear that a spined category $(\mathcal{C}, \Omega_n, \mathfrak{P})$ where $|\Omega_n| < n$ (resp. $\dot{\omega}(\Omega_n) > n$) does not admit any S-functors. To see this, suppose by way of contradiction that such a spined category did admit an S-functor F. By the definition of $|-|$, there must be an arrow $\Omega_n \to \Omega_m$ with $m < n$ (since we are assuming $|\Omega_n| < n$) and, since $F : \mathcal{C} \to \mathbb{N}_{\leq}$ is a functor, we must have that $F[\Omega_n \to \Omega_m] = (F[\Omega_n] \leq F[\Omega_m])$. However, since F is an S-functor it preserves the spine (Property **SC1**). Thus we have that $F[\Omega_m] = m$ and hence $F[\Omega_n] \leq m < n$ which violates Property **SC1**.

These observations show in particular that the spined category $(\mathbf{FinSet}, (\{1, 2, \ldots, n\})_{n\in\mathbb{N}}, +)$ (Example 3.3.4) has no S-functors (since every set can be mapped to a singleton set by a function). Similarly, notice

that, denoting by $\mathbf{Gr}^R_{homo}$ the category of *reflexive* graphs and graph-homomorphisms, there are no S-functors defined on $\mathbf{Gr}^R_{homo}$ since every graph has a homomorphism to K_1 (since this is a reflexive graph in this category). However, S-functors may fail to exist even if $|\Omega_n| = n$ (resp. $\dot{\omega}(\Omega_n) = n$). We construct such an example below.

Lemma 3.3.10. *Let $(\mathcal{C}, \Omega, \mathfrak{P})$ be a spined category. If there exists a span $\Omega_L \xleftarrow{\ell} \Omega_m \xrightarrow{r} \Omega_R$ in $\mathcal{C}$ such that the proxy-pushout $\mathfrak{P}(\ell, r)$ is isomorphic to Ω_n with $\max\{L, R\} < n$, then there is no S-functor over $(\mathcal{C}, \Omega, \mathfrak{P})$.*

Proof. By way of contradiction, suppose that there exists an S-functor F over $(\mathcal{C}, \Omega, \mathfrak{P})$; then

$$\begin{aligned} F[\Omega_n] &= F[\mathfrak{P}(\ell, r)] && \text{(since } \mathfrak{P}(\ell, r) \text{ is isomorphic to } \Omega_n) \\ &= \max\{F[\Omega_L], F[\Omega_R]\} && \text{(by Property SF2)} \\ &= \max\{L, R\} && \text{(by Property SF1)} \\ &< n. \end{aligned}$$

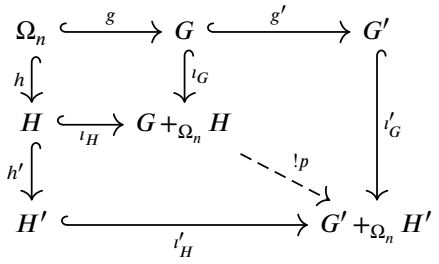
This contradicts the fact that F preserves the spine (i.e. Property SF1 is violated). ∎

Lemma 3.3.10 gives us another way of showing that $(\mathbf{FinSet}, (\{1, 2, \ldots, n\})_{n \in \mathbb{N}}, +)$ does not admit any S-functor; however, it can do much more: it allows us to give an example of a spined category which admits no S-functor despite satisfying the equation $|\Omega_n| = n = \dot{\omega}(\Omega_n)$ for all n.

Proposition 3.3.11. *There exist spined categories $(\mathcal{C}, \Omega, \mathfrak{P})$ satisfying $|\Omega_n| = n = \dot{\omega}(\Omega_n)$ that do not admit any S-functors.*

Proof. Consider the category $\mathbf{Poset}_{mono}$ which has finite posets as objects and order-preserving injections as morphisms. Let Ω_n denote set $\{m \in \mathbb{N} \mid m \leq n\}$ under its usual linear ordering, and let $\mathfrak{P}$ assign to each span of the form $G \xleftarrow{g} \Omega_n \xrightarrow{h} H$ the pushout $G +_{\Omega_n} H$ of the span in $\mathbf{Poset}_{homo}$ (note that the category of posets has all pushouts [5]). We will show that the triple $(\mathbf{Poset}_{mono}, \Omega, \mathfrak{P})$ forms a spined category that does not admit any S-functors.

Take any poset P on n elements and note that there is a monomorphism from P to Ω_n. This verifies Property SC1. For Property SC2 consider the following diagram.



Notice that the arrows $\iota_G, \iota'_G, \iota_H, \iota'_H$ are all monic. We have to establish that the morphism $p : G +_{\Omega_n} H \to G' +_{\Omega_n} H'$ (which is unique since it is a pushout arrow in $\mathbf{Poset}_{homo}$) is monic as well. Notice that p can be defined piece-wise as the map taking any point x in $G +_{\Omega_n} H$ to $(\iota'_G \circ g')(x)$ if x is in G and to $(\iota'_H \circ h')(x)$ otherwise. Since $G' +_{\Omega_n} H'$ is obtained by identifying the points in the image of Ω_n under $g' \circ g$ with the points in the image of Ω_n under $h' \circ h$, we have that, by its definition, p must be injective and hence monic.

Now we show that $(\mathbf{Poset}_{mono}, \Omega, \mathfrak{P})$ does not admit any S-functors. Consider the linearly ordered posets $\Omega_3 = \{a \leq b \leq c\}$, $\Omega_2 = \{d \leq e\}$, and $\Omega_1 = \{x\}$ and the span $\Omega_2 \xhookleftarrow{\ell} \Omega_1 \xhookrightarrow{r} \Omega_3$ given by the monomorphisms $\ell(x) = d$ and $r(x) = c$. By Lemma 3.3.10, this concludes the proof since the proxy-pushout of f, g is isomorphic to Ω_4. ■

Instead of exhaustively enumerating all possible obstructions to the existence of S-functors, we restrict our attention to those spined categories that come equipped with at least one S-functor. We shall see that the existence of a single S-functor already suffices to construct a functorial analogue of tree-width on any such category.

Definition 3.3.12. We call a spined category *measurable* if it admits at least one S-functor.

Of course **Nat** is a measurable spined category (where measurability is witnessed by the indentity functor $\mathbb{1}_{\mathbf{Nat}} : \mathbf{Nat} \to \mathbf{Nat}$). The measurability of $\mathbf{Gr}_{mono}$ follows from Proposition 3.3.8 or alternatively by noticing that the clique number is an S-functor. One might then naturally conjecture the generalized clique number to always be an S-functor. However, as we prove below, this is not the case.

Proposition 3.3.13. *The generalized clique number $\dot{\omega}$ need not give rise to an S-functor over an arbitrary measurable spined category.*

Proof. Equip the natural numbers with the divisibility relation, and regard the resulting poset as a category $\mathbb{N}_{div}$. Equip $\mathbb{N}_{div}$ with the spine

$$\Omega_n = \prod_{p \leq n} p^n$$

where p ranges over the primes. The poset category $\mathbb{N}_{div}$ has all pushouts: the pushout of objects n, m is given by the least common multiple of n and m. Let $\mathfrak{P}(x \to n, x \to m)$ denote the least common multiple $\text{lcm}(n, m)$. We verify each of the spined category properties in turn:

SC1: Take any $n \in \mathbb{N}$. Let p be the largest prime appearing in the prime factorization of n and let k be the largest exponent appearing in the prime factorization of n. Then n divides Ω_{p^k}.

SC2: Immediate from Proposition 3.3.2.

Consider the map ϵ that sends each object $n \in \mathbb{N}_{div}$ to the highest exponent that occurs in the prime factorization of n (this is the sequence OEIS A051903 [60]). This map satisfies Property **SF1** since the highest exponent that occurs in the prime factorization of $\Omega_n = \prod_{p \leq n \text{ and } p \text{ prime}} p^n$ is n. Furthermore ϵ satisfies Property **SF2** since the highest exponent occurring in the factorization of $\mathfrak{P}(x \to n, x \to m) = \text{lcm}(n, m)$ clearly is $\max\{\epsilon(n), \epsilon(m)\}$. Thus ϵ is an S-functor on the category $(\mathbb{N}_{div}, \Omega, \text{lcm})$ (which is therefore a measurable spined category). However, we claim that the generalized clique-number $\dot{\omega}$ is not an S-functor on this spined category.

To see this, consider the objects 16 and 81 in $\mathbb{N}_{div}$. Since $\Omega_2 = 2^2 = 4$ and $\Omega_3 = 2^3 \cdot 3^3 = 216$, the largest n for which Ω_n divides 16 is $\dot{\omega}[16] = 2$. Similarly, $\dot{\omega}[81] = 1$. However, we have $\dot{\omega}[\text{lcm}(16, 81)] = \dot{\omega}[1296] = \dot{\omega}[\Omega_4] = 4 \neq 2$. ■

Unlike the generalized clique number, the order map *does* give rise to an S-functor over the category $(\mathbb{N}_{div}, (\prod_{p \leq n \text{ and } p \text{ prime}} p^n)_{n \in \mathbb{N}}, \text{lcm})$.

Proposition 3.3.14. *The order map is an S-functor over* $(\mathbb{N}_{div}, (\prod_{p \leq n \text{ and } p \text{ prime}} p^n)_{n \in \mathbb{N}}, \text{lcm})$.

Proof. First we shall show that $|-|$ satisfies Property **SF2**. Since n and m both divide $\text{lcm}(n, m)$, if $|\text{lcm}(n, m)| = k$, then both n and m would divide Ω_k as well; thus $\max\{|n|, |m|\} \leq |\text{lcm}(n, m)|$. For the other direction, notice that, if $|n| = n'$ and $|m| = m'$ with $n' \geq m'$ (w.l.o.g.), then both n and m divide $\Omega_{n'} = \prod_{p \leq n' \text{ and } p \text{ prime}} p^{n'}$ (i.e. it is a common multiple) and hence $\text{lcm}(n, m)$ divides $\Omega_{n'}$ as well (i.e. $|\text{lcm}(n, m)| \leq \max\{|n|, |m|\}$). For Property **SF1**, since every integer divides itself (and since $\Omega_{n'} < \Omega_n$ for all $n' < n$) we have that $|\Omega_n| = n$, as desired. ■

In general, however, the order map need not give rise to an S-functor.

Proposition 3.3.15. *The order map $X \mapsto |X|$ need not give rise to an S-functor over an arbitrary measurable spined category.*

Proof. We claim that the order map does not constitute an S-functor over the measurable spined category $\mathbf{Gr}_{mono}$. To see this, note that, if $X \mapsto |V(X)|$ were an S-functor, then it would preserve proxy-pushouts; in particular we would have $3 = |P_3| = |K_2 \#_{K_1} K_2| = \max\{|K_2|, |K_2|\} = 2$. ■

3.4 Tree-width in a measurable spined category

In this section we give an abstract analogue of tree-width in our categorial setting, by proving a theorem in the style of Halin's theorem (Theorem 3.1.2). To do so, we must find a maximum S-functor under the point-wise order. An obvious candidate is the map taking every object to its order (Definition 3.3.9). However, as we just saw (Proposition 3.3.15), the order need not constitute an S-functor for measurable spined categories. Thus, rather than trying to define an S-functor via morphisms from objects to elements of the spine, we will consider morphisms to elements of a distinguished class of objects which we call *pseudo-chordal*. These objects will be used to define our abstract analogue of tree-width as an S-functor on any measurable spined category. We will conclude the section by showing how our abstract characterization of tree-width allows us to recover the familiar notions of graph and hypergraph tree-width.

Definition 3.4.1. We call an object X of a spined category $(\mathcal{C}, \Omega, \mathfrak{P})$ *pseudo-chordal* if for every two S-functors $F, G : \mathcal{C} \to \mathbf{Nat}$ we have $F[X] = G[X]$

Note that, if the spined category is not measurable, then every object is trivially pseudo-chordal.

Proposition 3.4.2. *The set Q of all pseudo-chordal objects of a spined category $(\mathcal{C}, \Omega, \mathfrak{P})$ contains all objects of the form Ω_n, and is closed under proxy-pushouts in the following sense: given two objects $A, B \in Q$ and any span $A \xleftarrow{f} \Omega_n \xrightarrow{g} B$, we always have $\mathfrak{P}(f, g) \in Q$.*

Proof. Given two S-functors F, G on $\mathcal{C}$, we always have $F[\Omega_n] = n = G[\Omega_n]$ since S-functors preserve the spine (Property **SF1**). Moreover, by Property **SF2**, S-functors preserve proxy-pushouts. This means that, since the proxy-pushout in **Nat** is the max operator, given any two pseudo-chordal objects $A, B \in Q$ and any span $A \xleftarrow{a} \Omega_n \xrightarrow{b} B$, we have

$$
\begin{aligned}
F[\mathfrak{P}(a,b)] &= \max\{F[A].F[B]\} && (\text{ by Property } \mathbf{SF2})\\
&= \max\{G[A].G[B]\} && (\text{since } A \text{ and } B \text{ are pseudo-chordal})\\
&= G[\mathfrak{P}(a,b)] && (\text{ by Property } \mathbf{SF2}).
\end{aligned}
$$

■

In light of Proposition 3.4.2, it is natural to distinguish the smallest set of pseudo-chordal objects that contains the spine and which is closed under proxy-pushouts. We call this the set of *chordal objects*. The name is given in analogy to chordal graphs, a resemblance that is best seen in the following recursive definition of chordal objects.

Definition 3.4.3. We define the set of *chordal objects* of the category spined category $\mathcal{C}$ inductively, as the smallest set S of objects satisfying the following:

- $\Omega_n \in S$ for all $n \in \mathbb{N}$, and
- $\mathfrak{P}(a,b) \in S$ for all objects $A, B \in S$ and spans $A \xleftarrow{a} \Omega_n \xrightarrow{b} B$.

Note that the notions of chordality and pseudo-chordality are well-defined even in *non-measurable* categories (since every object is pseudo-chordal if the category in question is not measurable).

Example 3.4.4. *Notice that all objects of* **Nat** *are spinal and hence they are all also chordal.*

Another example of a spined category in which every chordal object is also spinal is given by $\mathbb{N}_{div}$ (although, in contrast to the previous example, not every element of this category is in the spine).

Example 3.4.5. *Consider the spined category* $(\mathbb{N}_{div}, (\prod_{p\leq n \text{ and } p \text{ prime}} p^n)_{n\in\mathbb{N}}, \mathrm{lcm})$. *It is measurable (by Proposition 3.3.14) and all chordal objects are also spinal since*

$$\mathrm{lcm}\Big(\prod_{p\leq n \text{ and } p \text{ prime}} p^n, \prod_{p\leq m \text{ and } p \text{ prime}} p^m\Big) = \prod_{p\leq \max\{n,m\} \text{ and } p \text{ prime}} p^{\max\{n,m\}}.$$

Notice, however, that most often the set of chordal objects is a strict superset of the set of all spinal objects. For example, consider the familiar category $\mathbf{Gr}_{mono}$ of graphs and injective graph homomorphisms.

Example 3.4.6. *By comparing to the Defintion 1.2.1 (i.e. Dirac's Theorem for chordal graphs), we immediately see that, for any object X in the spined category $(\mathbf{Gr}_{mono}, (K_n)_{n\in\mathbb{N}}, \mathfrak{P})$ (recall that $\mathfrak{P}$ takes spinal spans to clique-sums), X is chordal (in the sense of Definition 3.4.3) if and only if X is a chordal graph.*

As an immediate consequence of Proposition 3.4.2 we have the following result.

Corollary 3.4.7. *All chordal objects are pseudo-chordal.*

However, note that the converse of Corollary 3.4.7 does not hold; as we shall see, it fails even in $\mathbf{Gr}_{mono}$.

Proposition 3.4.8. *Pseudo-chordality does not imply chordality.*

Proof. We will show that, in the spined category $\mathbf{Gr}_{mono}$, there exists a non-chordal object for which every pair of S-functors agree. To this end, consider, for some $n \in \mathbb{N}$, the object $K_n \#_{K_1} C_n$ obtained by identifying a vertex of an n-clique to a vertex of an n-cycle. The graph $K_n \#_{K_1} C_n$ is clearly not chordal for $n > 3$ and hence it is not a chordal object in this category (recall – by Example 3.4.6 – that the two notions of chordality coincide in this spined category). We claim, however, that $K_n \#_{K_1} C_n$ is pseudo-chordal. To see this, notice that, since C_n is a subgraph of K_n, we have a sequence of injective graph homomorphisms

$$K_n \hookrightarrow K_n \#_{K_1} C_n \hookrightarrow K_n \#_{K_1} K_n.$$

Thus, for any S-functor F, we have

$$n = F[K_n] \leq F[K_n \#_{K_1} C_n] \leq F[K_n \#_{K_1} K_n] = \max\{F[K_n], F[K_n]\} = n.$$

■

We will use pseudo-chordal objects to define the notion of a *pseudo-chordal completion* of an object of a spined category. We point out that the name was given in analogy to the operation of a chordal completion of graphs (i.e. the addition of a set F of edges to some graph G such that the resulting graph $(V(G), E(G) \cup F)$ is chordal).

Definition 3.4.9. A *pseudo-chordal completion* of an object X of a spined category $(\mathcal{C}, \Omega, \mathfrak{P})$ is an *arrow* $\delta : X \hookrightarrow H$ for some *pseudo-chordal* object H. If the pseudo-chordal object H is also chordal, then we call δ a *chordal completion*.

Recall that, for graphs, one can define the tree-width a graph G as:

$$\mathbf{tw}(G) = \min\{\omega(H) - 1 : H \text{ chordal completion of } G\}.$$

Notice that this is simply Equation 1.1 restated; ω is the clique number (in the graph-theoretic sense). With this in mind, observe that the following definition of the *width of a pseudo-chordal completion* furthers the analogy between our construction and the tree-width of graphs.

Definition 3.4.10. Let X and F be respectively an object and an S-functor in some measurable spined category. The *width* of a pseudo-chordal (resp. chordal) completion $\delta : X \hookrightarrow H$ of X is the value $F[H]$.

We point out that, in contrast to the case of graphs, we do not define the width of a pseudo-chordal completion by using the generalized clique number $\dot{\omega}$. This is because $\dot{\omega}$ need not be an S-functor in general (by Proposition 3.3.13). We instead use *any* S-functor F (for clarity we note that the choice of F in Definition 3.4.10 is inconsequential since – by the definition of pseudo-chordality – every two S-functors agree on pseudo-chordal objects).

Proposition 3.4.11. *Let $(\mathcal{C}, \Omega, \mathfrak{P})$ be a measurable spined category and denote by $\Delta[X]$ and $\Delta^{ch}[X]$ the minimum possible width of respectively any pseudo-chordal completion of the object X and any chordal completion of X. Then Δ and Δ^{ch} are functors from $\mathcal{C}$ to $\mathbb{N}_{\leq}$.*

Proof. We only prove the claim for Δ since the argument for Δ^{ch} is the same. Let F be any S-functor over $(\mathcal{C}, \Omega, \mathfrak{P})$. We need to verify that, for every arrow $f : X \to Y$ in $\mathcal{C}$, we have $\Delta[X] \leq \Delta[Y]$. To this end, for any such arrow $f : X \to Y$, take two minimum-width pseudo-chordal completions δ_X and δ_Y of X and Y respectively as in the following diagram.

$$\begin{array}{ccc} X & \xrightarrow{\quad f \quad} & Y \\ {\scriptstyle \delta_X}\downarrow & & \downarrow{\scriptstyle \delta_Y} \\ H_X & & H_y \end{array} \qquad \text{(where } H_X \text{ and } H_Y \text{ are pseudo-chordal)}$$

Notice that the composite arrow $\delta_Y \circ f$ is also a pseudo-chordal completion of X. Thus, by the minimality of the width of δ_X, we have $\Delta[X] = F[H_X] \leq F[H_Y] = \Delta[Y]$, as desired. ∎

Definition 3.4.12. Let Δ and Δ^{ch} be the functors defined in Proposition 3.4.11. We call Δ the *triangulation functor* and Δ^{ch} the *chordal triangulation functor*.

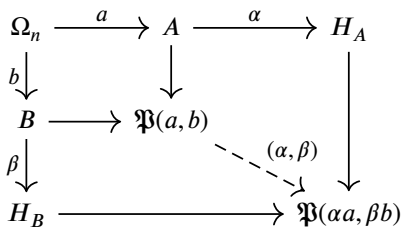
Our goal now is to show that the triangulation functor of a measurable spined category is in fact an S-functor. Specifically we prove our main theorem which states that both Δ and Δ^{ch} are S-functors in any measurable spined category.

Theorem 3.4.13. *Both the triangulation and chordal-triangulation functors are S-functors in any measurable spined category.*

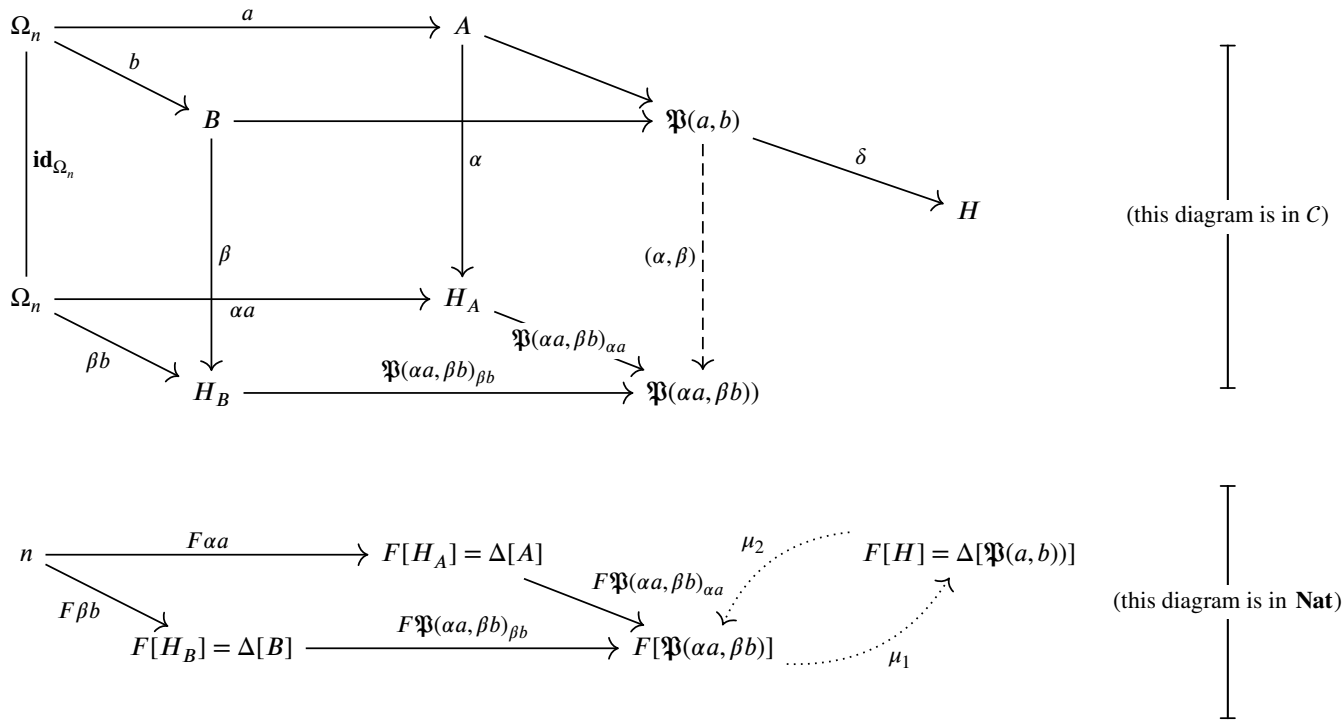
Proof. Let $(C, \Omega, \mathfrak{P})$ be any measurable spined category equipped with some S-functor F. We will prove the statement only for Δ since the method of proof for the Δ^{ch} case is the same.

Consider a pseudo-chordal completion $c : X \to H$ of a pseudo-chordal object X. Then $F[X] \leq F[H]$, and so the identity pseudo-chordal completion of X has has minimum width among all pseudo-chordal completions of X. This proves that $\Delta[\Omega_n] = n$ and hence that Δ satisfies property **SF1**.

For **SF2**, consider any span $A \xleftarrow{a} \Omega_n \xrightarrow{b} B$ in C. We have to prove that $\Delta[\mathfrak{P}(a,b)] = \max\{\Delta[A], \Delta[B]\}$. Choose a pseudo-chordal completion $\alpha : A \to H_A$ (resp. $\beta : B \to H_B$) for which $F[H_A]$ (resp. $F[H_B]$) is minimal. Using property **SC2**, there is a unique arrow $(\alpha, \beta) : \mathfrak{P}(a,b) \to \mathfrak{P}(\alpha a, \beta b)$ such that the following diagram commutes.



Now take a pseudo-chordal completion $\delta : \mathfrak{P}(a,b) \to H$ of $\mathfrak{P}(a,b)$ for which the quantity $F[H]$ is minimal. Consider the following diagram (recall that we already deduced the existence of the *dashed* arrow (α, β) in the previous paragraph).



Note that, since F is an S-functor, the bottom square of the diagram above (which is a diagram in **Nat**) commutes and thus

$$\begin{aligned} F[\mathfrak{P}(\alpha a,\beta b)] &= \max\{F[H_A],F[H_B]\} && (\text{ since } F \text{ satisfies Property SF2})\\ &= \max\{\Delta[A],\Delta[B]\} && (\text{ by the definition of } \Delta). \end{aligned}$$

Consequently, to show that $\Delta[\mathfrak{P}(a,b)]=\max\{\Delta[A],\Delta[B]\}$ (i.e. to show that Δ satisfies Property SF2), it suffices to show that $\Delta[\mathfrak{P}(a,b)]=F[\mathfrak{P}(\alpha a,\beta b)]$. In particular, this amounts to deducing the existence of the dotted arrows μ_1 and μ_2 in the diagram above (recall that, in **Nat**, an arrow $n\to n'$ corresponds to the inequality $n\le n'$).

To show the existence of μ_1, notice that, since $\delta\circ\mathfrak{P}(a,b)_a$ constitutes a pseudo-chordal completion of A and since we chose H_A so that $F[H_A]$ is minimal, we have $F[H_A]\le F[H]$. Similarly we can deduce $F[H_B]\le F[H]$. Thus we have

$$F[\mathfrak{P}(\alpha a,\beta b)]=\max\{F[H_A],F[H_B]\}\le F[H],$$

which proves the existence of μ_1.

To show the existence of μ_2, recall, by Proposition 3.4.2, that we know that the set of pseudo-chordal objects is closed under proxy-pushouts. Since H_A and H_B are pseudo-chordal, so is their proxy-pushout $\mathfrak{P}(\alpha a, \beta b)$. Hence $(\alpha, \beta) : \mathfrak{P}(a,b) \to \mathfrak{P}(\alpha a, \beta b)$ is a pseudo-chordal completion of $\mathfrak{P}(a,b)$. However, so is H. In fact we chose H so that $F[H]$ is minimal (since $F[H] = \Delta[\mathfrak{P}(a,b)]$). Thus we have $F[\mathfrak{P}(\alpha a, \beta b)] \geq F[H]$, which proves the existence of μ_2. ■

Surprisingly (especially given the fact that, by Proposition 3.4.8, pseudo-chordality does *not* imply chordality) the S-functors Δ and Δ^{ch} defined above coincide.

Corollary 3.4.14. *In any measurable spined category we have $\Delta = \Delta^{ch}$.*

Proof. Consider any measurable spined category $(\mathcal{C}, \Omega, \mathfrak{P})$ equipped with an S-functor F and let X be an object in $\mathcal{C}$. Since every chordal object is also pseudo-chordal (Corollary 3.4.7) we know that $\Delta[X] \leq \Delta^{ch}[X]$. We now show that given any minimum-width pseudo-chordal completion $\delta : X \to H$ of X, we can find a chordal completion of X of the same width as δ.

Let $\gamma : H \to H^{ch}$ be a minimum-width chordal completion of H. Since H is pseudo-chordal, all S-functors take the same value on H. In particular this means that $\Delta[H] = \Delta^{ch}[H]$ since both Δ and Δ^{ch} are S-functors by Theorem 3.4.13. Thus we have $F[H] = \Delta[H] = \Delta^{ch}[H] = F[H^{ch}]$. But then $\gamma \circ \delta$ is a chordal completion of X with width $F[H^{ch}] = F[H]$, as desired. ■

The triangulation S-functor Δ satisfies a maximality property broadly analogous to Theorem 3.1.2 (recall that this theorem states – among other facts – that tree-width is the maximal element out of all of Halin's S-functions).

Theorem 3.4.15. *Let $(\mathcal{C}, \Omega, \mathfrak{P})$ be any measurable spined category. The set of all S-functors over $(\mathcal{C}, \Omega, \mathfrak{P})$ is a join semi-lattice under the pointwise ordering with Δ as its maximum element.*

Proof. Let $\mathcal{Z}$ be any non-empty (possibly infinite) subset of the set of S-functors over $(\mathcal{C}, \Omega, \mathfrak{P})$. In what follows we shall first construct the supremum of $\mathcal{Z}$ and then we will prove that it constitutes an S-functor.

Define the map $F_{\mathcal{Z}} : \mathcal{C} \to \mathbb{N}$ for any W in $\mathcal{C}$ as $F_{\mathcal{Z}}[W] := \max_{F' \in \mathcal{Z}} F'[W]$. (Note that this maximum always exists since every object X is mapped by any S-functor to at most the value of $|X|$ and hence $\{F'[X] : F' \in \mathcal{Z}\}$ is a bounded set of integers.)

We claim that, for any arrow $m : X \to Y$ in $\mathcal{C}$, we have $F_{\mathcal{Z}}[X] \leq F_{\mathcal{Z}}[Y]$. To see this, let Q be an element of $\mathcal{Z}$ such that $Q[X] = F_{\mathcal{Z}}[X]$ (by the definition of $F_{\mathcal{Z}}$ and since $\mathcal{Z}$ is non-empty, such a Q always exists). The functoriality of Q implies that, if there is an arrow $X \to Y$ in $\mathcal{C}$, then $Q[X] \leq Q[Y]$; in particular we can deduce that

$$F_{\mathcal{Z}}[X] = Q[X] \leq Q[Y] \leq \max_{F' \in \mathcal{Z}} F'[Y] = F_{\mathcal{Z}}[Y].$$

Hence there is an arrow $g : F_{\mathcal{Z}}[X] \to F_{\mathcal{Z}}[Y]$ in **Nat**, which means that we can (slightly abusing notation) render $F_{\mathcal{Z}}$ a functor by extending the definition of $F_{\mathcal{Z}}$ to map any arrow $m : X \to Y$ to the arrow $g : F_{\mathcal{Z}}[X] \to F_{\mathcal{Z}}[Y]$ in **Nat**.

From what we just showed (and given the extension to the definition of $F_{\mathcal{Z}}$ that we described above), we know that $F_{\mathcal{Z}}$ is a functor. Now we will show that it is a spined functor. Note that $F_{\mathcal{Z}}$ clearly preserves the spine; furthermore, for any span $A \xleftarrow{a} \Omega_n \xrightarrow{b} B$, we have

$$\begin{aligned} F_{\mathcal{Z}}[\mathfrak{P}(a,b)] &= \max_{F' \in \mathcal{Z}} F'[\mathfrak{P}(a,b)] && \text{(by the definition of } F_{\mathcal{Z}}) \\ &= \max_{F' \in \mathcal{Z}} \max\{F'[A], F'[B]\} && \text{(since } F' \text{ is an S-functor)} \\ &= \max\{F_{\mathcal{Z}}[A], F_{\mathcal{Z}}[B]\}. \end{aligned}$$

Thus $F_{\mathcal{Z}}$ is an S-functor since it satisfies Properties **SF1** and **SF2**. In particular we proved that the set of all S-functors over $(\mathcal{C}, \Omega, \mathfrak{P})$ is a join semi-lattice under the point-wise ordering.

To show that Δ is the largest element of this semi-lattice, we need to prove that, given any S-functor F, we have $F[X] \leq \Delta[X]$ for all objects X. To this end, take any pseudo-chordal completion $\delta : X \to H$ of some object X. For any S-functor F, the following diagram commutes (by the functoriality of F).

$$\begin{array}{ccc} X & \xrightarrow{\delta} & H \\ {\scriptstyle F}\downarrow & & \downarrow{\scriptstyle F} \\ F[X] & \xrightarrow[F_\delta]{} & F[H] \end{array}$$

But since $\Delta[X] := F[H]$, we have $F[X] \leq \Delta[X]$ and hence – as desired – Δ is the maximum element of the join semi-lattice of all S-functors. ■

Abstract analogue of tree-width We will now given further justification of our claim that the triangulation functor Δ constitutes an abstract analogue of tree-width: we will show that, when instantiated on either graphs

or hypergraphs, we recover the appropriate notions of tree-width; i.e. we will show that, in these categories, we have $\Delta = \mathbf{tw}+1$.

Earlier we showed (Proposition 3.3.8) that every S-function yields an S-functor over $\mathbf{Gr}_{mono}$; thus we already know that $\mathbf{tw}+1$ is an S-functor (since it is an S-function by Theorem 3.1.2). However, we have not yet shown that $\Delta = \mathbf{tw}+1$; we prove this below.

Corollary 3.4.16. *Let Δ be the triangulation functor of $\mathbf{Gr}_{mono}$. Then, for any graph G, we have $\Delta[G] = \mathbf{tw}(G)+1$.*

Proof. Recall that an object of $\mathbf{Gr}_{mono}$ is chordal if and only if it is a chordal graph (by Example 3.4.6). Hence we simply compute:

$$\begin{aligned}
\mathbf{tw}(G)+1 &= \min\{\omega(H) : H \text{ is a chordal completion of } G\} && \text{(recall, this is Equation 1.1)}\\
&= \min\{\dot{\omega}(H) : H \text{ is a chordal completion of } G\} && (\text{since } \dot{\omega} \text{ and } \omega \text{ agree in } \mathbf{Gr}_{mono})\\
&= \Delta^{ch}[G] && (\text{since } \dot{\omega} \text{ is an S-functor in } \mathbf{Gr}_{mono})\\
&= \Delta[G] && \text{(by Corollary 3.4.14).}
\end{aligned}$$

■

Next we consider the category $\mathbf{HGr}_{mono}$ of hypergraphs and their injective homomoprhisms which we describe now. Let H_1 and H_2 be hypergraphs; a vertex map $h : V(H_1) \to V(H_2)$ is a *hypergraph homomorphism* if it preserves hyper-edges; that is to say that, for every edge $F \in E(H_1)$, the set $h(F) := \{h(x) : x \in F\}$ is a hyper-edge in H_2. Hypergraph homomorphisms clearly compose associatively, thus we can define the category $\mathbf{HGr}_{mono}$ which has finite hypergraphs as objects and injective hypergraph homomorphisms as arrows.

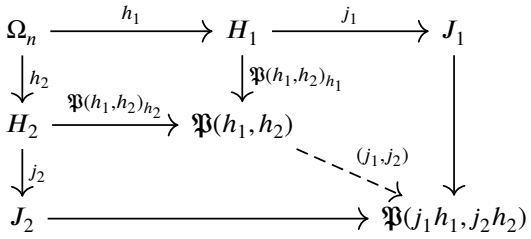
Theorem 3.4.17. *Let $\Omega : \mathbb{N}_{=} \to \mathbf{HGr}_{mono}$ be the functor taking every integer n to the hypergraph $([n], 2^{[n]})$ and let $\mathfrak{P}$ assign to each span of the form $H_1 \xleftarrow{h_1} \Omega_n \xrightarrow{h_2} H_2$ in $\mathbf{HGr}_{mono}$ the cospan*

$$H_1 \xrightarrow[\mathfrak{P}(h_1,h_2)_{h_1}]{} \mathfrak{P}(h_1,h_2) \xleftarrow{\mathfrak{P}(h_1,h_2)_{h_2}} H_2$$

where $\mathfrak{P}(h_1,h_2) := \Big(V(H_1) +_{V(\Omega_n)} V(H_2),\ \big(E(H_1) \uplus E(H_2)\big)/_{h_1=h_2}\Big)$ (note that $V(H_1) +_{V(\Omega_n)} V(H_2)$ is a pushout in $\mathbf{FinSet}$) and $\mathfrak{P}(h_1,h_2)_{h_i}$ is the injective hypergraph homomorphisms given by the vertex

map $V(H_i) \hookrightarrow V(H_1) +_{V(\Omega_n)} V(H_2)$ in **FinSet** *(i.e. $\mathfrak{P}(h_1, h_2)_{h_i}$ is the obvious injective hypergraph homomorphism witnessing that H_i is a sub-hypergraph of $\mathfrak{P}(h_1, h_2)$). Then the triple $(\mathbf{HGr}, \Omega, \mathfrak{P})$ is a spined category.*

Proof. Clearly Property **SC1** is satisfied, so, to show Property **SC2**, consider the following diagram in $\mathbf{HGr}_{mono}$ (where, in what follows, we will argue for the existence and uniqueness of the arrow (j_1, j_2)).



We define $(j_1, j_2) : \mathfrak{P}(h_1, h_2) \to \mathfrak{P}(j_1h_1, j_2h_2)$ as

$$(j_1, j_2)(x) := \begin{cases} j_1(x) \text{ if } x \in V(H_1) \cap \mathfrak{P}(h_1, h_2) \\ j_2(x) \text{ otherwise.} \end{cases}$$

To see that (j_1, j_2) is the unique injective *vertex-map* making the diagram commute, consider the forgetful functor $V : \mathbf{Gr}_{mono} \to \mathbf{FinSet}$ taking every hypergraph to its vertex-set. By our definition of the proxy-pushout $\mathbf{HGr}_{mono}$, it follows that V maps proxy-pushouts in $\mathbf{HGr}_{mono}$ to pushouts in **FinSet**. Thus, we deduce both the existence and uniqueness of (j_1, j_2) from the definition of a pushout (Definition 3.2.6). All that remains to be shown is that the vertex-map (j_1, j_2) is a hypergraph homomorphism, but this follows immediately (similarly to the graph case in Proposition 3.2.7) from the definition of the proxy-pushout in $\mathbf{HGr}_{mono}$. ■

We will now show that, when instantiated over $\mathbf{HGr}_{mono}$, the triangulation functor agrees with the map $H \to \mathbf{tw}(H) + 1$. Rather than showing this directly (to do this the argument would be the similar to the one in Corollary 3.4.16), we will prove a more general result will will allow us to also deduce that there are uncountably-many S-functors over the category $\mathbf{HGr}_{mono}$.

To this end, note that we can construct a spined functor from the spined category $\mathbf{HGr}_{mono}$ of hypergraphs to the spined category $\mathbf{Gr}_{mono}$ of graphs. We do this by observing that the mapping

$\mathfrak{G} : \mathbf{HGr} \to \mathbf{Gr}_{mono}$ such that

$\mathfrak{G} : H \mapsto \big(V(H), \{\{x, y\} \subseteq V(H) : x \neq y \text{ and } \exists e \in E(H) \text{ with } \{x, y\} \subseteq e\}\big)$

which associates every hypergraph to its Gaifman graph (sometimes also referred to as 'primal graph') is clearly functorial; we shall refer to $\mathfrak{G}$ as the *Gaifman graph functor*.

Proposition 3.4.18. *The Gaifman graph functor* $\mathfrak{G} : \mathbf{HGr} \to \mathbf{Gr}_{mono}$ *is a spined functor.*

Proof. Note that $\mathfrak{G}$ preserves the spine since $\mathfrak{G}[([n], 2^{[n]})] \cong K_n$ (i.e. $\mathfrak{G}$ satisfies Property **SF1**). Now take the proxy-pushout $\mathfrak{P}(h_1, h_2)$ of some span $H_1 \xleftarrow{h_1} \Omega_n \xrightarrow{h_2} H_2$ in $\mathbf{HGr}_{mono}$. Recall that $\mathfrak{P}(h_1, h_2)$ is constructed by identifying H_1 and H_2 along $\Omega_n := ([n], 2^{[n]})$. Thus, since $\mathfrak{G}$ preserves the spine (as we just showed) we know that the Gaifman graph $\mathfrak{G}[\mathfrak{P}(h_1, h_2)]$ of $\mathfrak{P}(h_1, h_2)$ is given by the clique-sum along a K_n of the Gaifman graphs of H_1 and H_2. In other words we have $\mathfrak{G}[\mathfrak{P}(h_1, h_2)] = \mathfrak{G}[H_1] \#_{\mathfrak{G}[\Omega_n]} \mathfrak{G}[H_1]$ which proves that $\mathfrak{G}$ preserves proxy-pushouts (i.e. it satisfies Property **SF2**). Thus $\mathfrak{G}$ is a spined functor. ■

Since the composition of spined functors is again a spined functor, by Propositions 3.3.8 and 3.4.18, every S-functor over $\mathbf{Gr}_{mono}$ gives rise to an S-functor of the form $\mathbf{HGr} \xrightarrow{\mathfrak{G}} \mathbf{Gr}_{mono} \longrightarrow \mathbf{Nat}$. Furthermore, since every one of Halin's S-functions gives rise to an S-functor (Proposition 3.3.8) and since there are uncountably many S-functions [55], we have just proven the following corollary.

Corollary 3.4.19. *The spined category* $\mathbf{HGr}_{mono}$ *is measurable; in particular there are uncountably many S-functors over* $\mathbf{HGr}_{mono}$.

Now consider any proxy-pushout $\mathfrak{P}(h_1, h_2)$ of a span $H_1 \xleftarrow{h_1} \Omega_n \xrightarrow{h_2} H_2$ in $\mathbf{HGr}_{mono}$. It follows (in much the same way as it does for graphs) that the hypergraph tree-width of $\mathfrak{P}(h_1, h_2)$ is the maximum of $\mathbf{tw}(H_1)$ and $\mathbf{tw}(H_2)$. Since, by the definition of hypergraph tree-width, we have $\mathbf{tw}(([n], 2^{[n]})) = n - 1$, it follows that, in $(\mathbf{HGr}, \Phi)$, $\Delta(K) = \mathbf{tw}(K) + 1$ for any chordal object K in $(\mathbf{HGr}, \Phi)$. Thus we have shown the following result.

Corollary 3.4.20. *If* Δ *is the triangulation functor of* $(\mathbf{HGr}, \Omega, \mathfrak{P})$*, then, for any hypergraph* H*,* $\Delta(H) = \mathbf{tw}(H) + 1$.

3.5 New Spined Categories from Old

The spined categories encountered so far came equipped with their "standard" notion of (mono)morphism: posets with monotone maps, graphs with graph homomorphsims, hypergraphs with hypergraph homomorphisms. In contrast, for a class S of combinatorial objects decorated with extraneous structure (such as colored or labeled graphs), the appropriate choice of morphism may be less obvious. In these cases, a "forgetful" function

$f : S \to \mathcal{C}$ from S to some spined category $\mathcal{C}$ allows us to study properties of S by studying properties of its image in $\mathcal{C}$.

It is straightforward to check that we can define a category $S_{\downarrow f}$ – which we call the *S-category induced by f* – by taking $\mathbf{Ob}(S_{\downarrow f}) := S$ and, for any two objects A and B in S, setting $\mathbf{Hom}_{S_{\downarrow f}}(A, B) := \mathbf{Hom}_{\mathcal{C}}(f(A), f(B))$.

It will be convenient to notice that – up to categorial isomorphism – $f^{-1}(X)$ (for any object X in the range of f) consists of only one object in $S_{\downarrow f}$. To see this, suppose f is not injective (otherwise there is nothing to show) and let $A, B \in S$ be elements of the set $f^{-1}(X)$. By the definition of $S_{\downarrow f}$, we know that $\mathbb{1}_X \in \mathbf{Hom}_{S_{\downarrow f}}(A, B)$ since $\mathbf{Hom}_{S_{\downarrow f}}(A, B) = \mathbf{Hom}_{\mathcal{C}}(A, B)$. Thus A and B are isomorphic in $S_{\downarrow f}$ since identity arrows are always isomorphisms.

Note that by the construction of $S_{\downarrow f}$, the function f actually constitutes a faithful and injective (on objects and arrows) functor from $S_{\downarrow f}$ to $\mathcal{C}$. The next result shows that if $\mathcal{C}$ is spined and if the range of f is sufficiently large, then we can chose a spine Ω^S and a proxy-pushout $\mathfrak{P}^S$ on $S_{\downarrow f}$ which turns $(S_{\downarrow f}, \Omega^S, \mathfrak{P}^S)$ into a spined category and $f : (S_{\downarrow f}, \Omega^S, \mathfrak{P}^S) \to (\mathcal{C}, \Omega, \mathfrak{P})$ into a spined functor.

Theorem 3.5.1. *Let $(\mathcal{C}, \Omega, \mathfrak{P})$ be a spined category, S be a set and $f : S \to \mathcal{C}$ be a function. If f is*

($S_{\downarrow f}$1) *surjective on the spine of $\mathcal{C}$ (i.e. $\forall n \in \mathbb{N}, \exists X \in S$ s.t. $f(X) = \Omega_n$) and such that*

($S_{\downarrow f}$2) *for every span $f(X) \xleftarrow{x} \Omega_n \xrightarrow{y} f(Y)$ in $\mathcal{C}$, there exists a distinguished element $Z_{x,y} \in S$ such that $f(Z_{x,y}) = \mathfrak{P}(x, y)$,*

then we can choose a functor $\Omega^S : \mathbb{N}_= \to S_{\downarrow f}$ and an operation $\mathfrak{P}^S$ such that

1. *$(S_{\downarrow f}, \Omega^S, \mathfrak{P}^S)$ is a spined category and*
2. *f is a spined functor from $(S_{\downarrow f}, \Omega^S, \mathfrak{P}^S)$ to $(\mathcal{C}, \Omega, \mathfrak{P})$*
3. *if $(\mathcal{C}, \Omega, \mathfrak{P})$ is a measurable spined category, then so is $(S_{\downarrow f}, \Omega^S, \mathfrak{P}^S)$.*

Proof. Define Ω^S and $\mathfrak{P}^S$ as follows:

- $\Omega^S : \mathbb{N}_= \to S_{\downarrow f}$ is the functor taking each n to an element of $f^{-1}(\Omega_n)$ (we can think of this as picking a representative of the equivalence class $f^{-1}(\Omega_n)$ for each n since, as we observed earlier, all elements of $f^{-1}(\Omega_n)$ are isomorphic),

- $\mathfrak{P}^S$ is the operation assigning to each span $X \xleftarrow{x} \Omega_n \xrightarrow{y} Y$ in $S_{\downarrow f}$ the co-span

$$X \xrightarrow{\mathfrak{P}(x,y)_x} \mathfrak{P}^S(x,y) := Z_{x,y} \xleftarrow{\mathfrak{P}(g,h)_h} Y \ ,$$

where $Z_{x,y}$ is the distinguished element whose existence is guaranteed by the second property of f.

Now we will show that $(S_{\downarrow f}, \Omega^S, \mathfrak{P}^S)$ is a spined category. Property **SC1** holds in $(S_{\downarrow f}, \Omega^S, \mathfrak{P}^S)$ since it holds in $(C, \Omega, \mathfrak{P})$ and since, for all $A, B \in S$, we have $\mathbf{Hom}_{S_{\downarrow f}}(A,B) := \mathbf{Hom}_C(A,B)$. To show Property **SC2**, we must argue that that, for every diagram of the form

$$\begin{array}{ccccc}
Q \in f^{-1}(\Omega_n) & \xrightarrow{h_1} & H_1 & \xrightarrow{j_1} & J_1 \\
\downarrow{\scriptstyle h_2} & & \downarrow & & \downarrow \\
H_2 & \longrightarrow & \mathfrak{P}^S(h_1,h_2) & \overset{p}{\cdots\!\searrow} & \\
\downarrow{\scriptstyle j_2} & & & & \\
J_2 & & \longrightarrow & & \mathfrak{P}^S(j_1h_1, j_2h_2)
\end{array} \tag{3.1}$$

in $S_{\downarrow f}$ there is a unique arrow p (which is dotted in Diagram (3.1)) which makes the diagram commute.

By the second condition on f (i.e. Condition ($S_{\downarrow f}$**2**)), we know that $f(\mathfrak{P}^S(h_1,h_2)) = \mathfrak{P}(fh_1, fh_2)$ and $f(\mathfrak{P}^S(j_1h_1, j_2h_2)) = \mathfrak{P}(fj_1h_1, fj_2h_2)$. Thus we have that f maps Diagram (3.1) in $S_{\downarrow f}$ to the following diagram in C.

$$\begin{array}{ccccc}
\Omega_n & \xrightarrow{h_1} & f(H_1) & \xrightarrow{j_1} & f(J_1) \\
\downarrow{\scriptstyle h_2} & & \downarrow & & \downarrow \\
f(H_2) & \longrightarrow & f(\mathfrak{P}^S(h_1,h_2)) = \mathfrak{P}(fh_1, fh_2) & \overset{(j_1,j_2)}{\cdots\!\searrow} & \\
\downarrow{\scriptstyle j_2} & & & & \\
f(J_2) & & \longrightarrow & & f(\mathfrak{P}^S(j_1h_1, j_2h_2)) = \mathfrak{P}(fj_1h_1, fj_2h_2)
\end{array} \tag{3.2}$$

Since $(C, \Omega, \mathfrak{P})$ satisfies Property **SC2**, the dashed arrow (j_1, j_2) in Diagram (3.2) exists, is unique and makes the diagram commute. But, by the definition of $S_{\downarrow f}$, we have

$$\mathbf{Hom}_{S_{\downarrow f}}\big(\mathfrak{P}^S(h_1,h_2), \mathfrak{P}^S(j_1h_1, j_2h_2)\big) = \mathbf{Hom}_C\big(\mathfrak{P}(fh_1, fh_2), \mathfrak{P}(fj_1h_1, fj_2h_2))\big),$$

which means that $(j_1, j_2) \in \mathbf{Hom}_{S_{\downarrow f}}\big(\mathfrak{P}^S(h_1,h_2), \mathfrak{P}^S(j_1h_1, j_2h_2)\big)$. In other words we can simply take $p = (j_1, j_2)$, as desired.

Now we will argue that f is a spined functor. By the first condition on f (i.e. Condition ($S_{\downarrow f}$**1**)), we know that f preserves the spine. As we just argued in Diagrams (3.1) and (3.2) we know that f preserves *all* (by the second property of f; i.e. Condition ($S_{\downarrow f}$**2**)) proxy-pushouts (i.e. it satisfies Property **SF2** as well). Thus f is a spined functor from $(S_{\downarrow f}, \Omega^S, \mathfrak{P}^S)$ to $(\mathcal{C}, \Omega, \mathfrak{P})$.

Finally note that, since f is a spined functor from $(S_{\downarrow f}, \Omega^S, \mathfrak{P}^S)$ to $(\mathcal{C}, \Omega, \mathfrak{P})$, it must be that, if there exists an S-functor G over $(\mathcal{C}, \Omega, \mathfrak{P})$, then the composition $G \circ f$ is an S-functor over $(S_{\downarrow f}, \Omega^S, \mathfrak{P}^S)$. Thus $(S_{\downarrow f}, \Omega^S, \mathfrak{P}^S)$ is measurable whenever $(\mathcal{C}, \Omega, \mathfrak{P})$ is. ■

Theorem 3.5.1 allows us to easily define new spined categories from ones we already know. For example, denoting, for every graph G, the set of all functions of the form $f : V(G) \to \{1, \dots, |V(G)|\}$ as $\ell(G)$, consider the set $\mathcal{L} := \{\ell(G) : G \in \mathcal{G}\}$ of all vertex-labelings of all finite simple graphs. Let $Q : \mathcal{L} \to \mathbf{Gr}_{mono}$ be the surjection

$$Q : \big(f : G \to [|V(G)|]\big) \longmapsto G/_f$$

which takes every labeling $f : G \to [|V(G)|]$ in $\mathcal{L}$ to the quotient $G/_f$ of the graph G under f defined as

$$G/_f := (V(G)/_f, E(G)/_f \setminus \{xx : x \in V(G)\}).$$

Since $\mathbf{Gr}_{mono}$ is a measurable spined category, we know that $\mathcal{L}_{\downarrow Q}$ is also a measurable spined category (by applying Theorem 3.5.1). In particular, the triangulation functor Δ_ℓ of $\mathcal{L}_{\downarrow Q}$ takes any object $f : G \to [|V(G)|]$ in $\mathcal{L}_{\downarrow Q}$ to the tree-width of $G/_f$ plus 1.

This construction might seem peculiar, since it maps labeling functions (as opposed to graphs themselves) to tree-widths of quotiented graphs. Thus we define the *ℓ-tree-width of any graph G*, denoted $\mathbf{tw}_\ell(G)$, as $\mathbf{tw}_\ell(G) = \min_{f \in \ell(G)} \Delta_\ell[f]$. This becomes trivial if we allow all possible vertex-labelings (since we can obtain a K_1 as a quotient of any graph: simply label all vertices with the same label). However, by imposing restrictions on the permissible labelings, we can obtain more meaningful width-measures on graphs. We briefly consider two examples to demonstrate this principle.

Example 3.5.2 (Modular tree-width)**.** *Recall that a vertex-subset X of a graph G is a called a* module *in G if, for all vertices $z \in V(G) \setminus X$, either z is adjacent to every vertex in X or $N(z) \cap X = \emptyset$. We*

call a labeling function $f : V(G) \to [|V(G)|]$ modular *if, for all* $i \in [|V(G)|]$*, the preimage* $f^{-1}(i)$ *of* i *is a module in* G*. Thus, denoting by* $\mathcal{M}$ *the set* $\mathcal{M} := \{\{\lambda : \lambda \text{ is modular labeling of } G\} : G \in \mathcal{G}\}$ *of all modular labelings, we obtain, as we did above, a spined category* $\mathcal{M}_{\downarrow Q}$*, where* Q *is the function taking each modular labeling to its corresponding modular quotient.*

Note that the triangualation number of $\mathcal{M}_{\downarrow Q}$ *maps every modular labeling to the tree-width of the corresponding modular quotient. Thus we can define* modular tree-width *which takes any graph* G *to the minimum tree-width possible over the set of all modular quotients of* G*.*

Example 3.5.3 (Chromatic tree-width)**.** *Denote the set of all proper colorings as*

$$\mathbf{col} := \{\{\lambda : \lambda \text{ is proper coloring of } G\} : G \in \mathcal{G}\}.$$

Then, as we just did in Example 3.5.2, we can study the spined category $\mathbf{col}_{\downarrow Q}$ *and its triangulation functor. Proceeding as before, this immediately yields the notion of* chromatic tree-width*.*

3.6 Further Questions

As we have seen, spined categories provide a convenient categorial setting for the study of classes of recursively decomposble objects.

Among spined categories, the measurable ones come equipped with a distinguished S-functor, the triangulation functor of Definition 3.4.12, which can be seen as a general counterpart to the graph-theoretic notion of tree-width, and which gives rise to an associated notion of completion/decomposition. Moreover, Theorem 3.4.13 shows that the only possible obstructions to measurability are the *generic* ones: if there is no obstruction so strong that it precludes the existence of *every* S-functor, there can be no further obstruction preventing the existence of the triangulation functor.

Since most settings have only one obvious choice of structure-preserving morphism (which often fixes the pushout construction as well), functoriality leaves the choice of an appropriate spine as the only degree of freedom. This makes spined categories an interesting alternative to other techniques for defining graph width measures, such as *layouts*[9] (which we used in Chapter 2 to define directed branch-width in Definition 2.3.1 and which have also been used for defining branch-width [87], rank-width [83], $\mathbb{F}_4$-width [64], bi-cut-rank-width [64]

[9] Sometimes referred to as 'branch decompositions' of symmetric submodular functions.

and min-width [93]), which rely on less easily generalized, graph-theory-specific notions of *connectivity*. Finding algebraic examples of spined categories and associated width measures remains a promising avenue for further work. In particular, as we move from combinatorial structures towards algebraic and order-theoretic ones, choosing a spine becomes an abundant source of technical questions. As an example of such as question, we propose the following concrete direction for future work.

Question. Consider the category $\mathbf{Poset}_{oe}$ which has finite posets as objects and order embeddings as morphisms, equipped with the usual pushout construction. Is there a sequence of objects $n \mapsto \Omega_n$ which makes $\mathbf{Poset}_{oe}$ into a *measurable* spined category?

4 | Interval-membership-width: dynamic programming on temporal graphs

4.1 Introduction

In this chapter we shift our focus to the growing field of temporal graphs. As we shall see in Sections 4.1.1 and 4.1.2, there are salient differences between static and temporal graph theory. In fact we will provide many examples of problems that are polynomial-time solvable for static graphs, but NP-hard of temporal graphs. In keeping with the theme of this thesis, we will then then study how to cope with algorithmic intractability on temporal graphs by seeking measures of recursive decomposability that we can exploit algorithmically. As we shall see, it will not be enough to simply restrict the underlying static structure of temporal graph. In fact this means that we will not be able to immediately apply the techniques we developed in the previous chapters to temporal graphs: instead we will need to take a step back to gain some understanding of what algorithmically exploitable *temporal* structure might look like. We will do this by studying the complexity of some natural exploration problems (such as deciding whether a temporal graph is temporally Euelerian) on temporal graphs. We will then introduce a new temporal width-measure called *interval-membership-width* which, as we will show, will be useful for the design of linear-time dynamic-programming algorithms on classes of temporal graphs of bounded interval-membership-width.

4.1.1 Background on temporal graphs

Temporal graphs crop up naturally when modelling systems such as human and animal proximity networks, human communication networks, collaboration networks, citation networks, economic networks

and neuro-scientific networks [22, 56, 79]. Particularly active application areas are biological, ecological and epidemiological networks [22, 57, 58] (where one might be interested in studying community formation or the spread of information or disease), distributed computing [22, 79] (for example in the study of delay-tollerant networks where one tries to ensure properties – such as connectivity – over time) and opportunistic mobility [22] (where there might be physically moving objects – such as busses, taxis, trains – which transmit information between each-other at limited distances). We note that, being a young field born form diverse application domains, it is no surprise that temporal graphs have been re-discovered (or re-defined) many times. This has lead to the same notion to be known under many different names such as: *temporal graphs*, *temporal networks*, *dynamic graphs*, *evolving graphs* and *time-varying graphs*.

Formal preliminaries A 'graph that changes with time' is quite a broad concept and there are many different perspectives that one might take on what this should mean. For instance, should the edges appear and disappear over time, or should the vertices be the ones changing with time? Also, should we assume time to be discrete or should we assume a time-continuum? Addressing these questions, we will introduce two models in what follows. The first – which we shall refer to as[1] $\mathbb{T}$*-temporal networks* [22] (Definition 4.1.1) – is due to Casteigts, Flocchini, Quattrocchi and Santoro [22]. The second – which we shall call *temporal graphs* – is due to Kempe, Kleinberg, and Kumar [66] (Definition 4.1.2). $\mathbb{T}$-temporal networks are more general than temporal graphs; they summarise in a single definition many of the different formalisms that have been proposed so far. Although we will not work with $\mathbb{T}$-temporal networks in this thesis, we take the time to define them because they convey a succinct impression of the diverse set of questions that the research community is interested in.

Definition 4.1.1. [[22]] Take $\mathbb{T}$ to be one of either $\mathbb{N}$ or $\mathbb{R}$. A $\mathbb{T}$*-temporal (directed) network* is a quintuple $(G, \rho_e, \eta_e, \rho_v, \eta_v)$ where G is a (directed) graph and ρ_e, η_e, ρ_v and η_v are functions of the following types:

$$\rho_e : E(G) \times \mathbb{T} \to \{\bot, \top\}, \qquad \eta_e : E(G) \times \mathbb{T} \to \mathbb{T},$$

$$\rho_v : V(G) \times \mathbb{T} \to \{\bot, \top\}, \qquad \eta_v : V(G) \times \mathbb{T} \to \mathbb{T}.$$

The functions ρ_e and ρ_v of Definition 4.1.1 are called the *edge-* and *vertex-presence* functions respectively: they indicate if an edge or vertex is present at any given time. The functions η_e and η_v are called the *edge-* and *vertex-latency* functions respectively: they indicate how long it takes – at any given time – to cross any given active edge (or process information at any given active vertex).

[1] this is not the terminology used in [22]; we choose it in order to avoid any confusion with Kempe, Kleinberg, and Kumar's model [66].

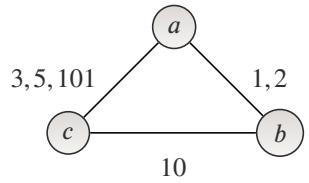


Figure 4.1: A temporal graph (K_3, τ) with lifetime 101.

For our purposes, we will only consider the discrete-time case (i.e. take $\mathbb{T} = \mathbb{N}$). In fact we will consider a much more restricted model in which the vertices do not vary with time and in which all notions of latency are disregarded. For many application domains, this is not a severe restriction since it can still be used to model many real-world problems ranging from epidemiology [37–39, 90] to social [7, 67] and trade networks [72] and distributed networks [65]. The formalism that we will adhere to in this thesis originates from the work of Kempe, Kleinberg, and Kumar [66] which is defined as follows (see also Figure 4.1). Note that, to distinguish it from $\mathbb{N}$-temporal networks, we will refer to this formalism as a *temporal graph*.

Definition 4.1.2. A *temporal graph* is a pair (G, τ) where G is a graph and $\tau : E(G) \to 2^{\mathbb{N}}$ is a function mapping edges of G to subsets of the naturals.

For any edge e in G, we call the set $\tau(e)$ the *time-set* of e (for example, given the temporal graph in Figure 4.1, the time-set of the edge ac is $\{2, 5, 101\}$). For any time $t \in \tau(e)$ we say that e is *active* at time t and we call the pair (e, t) a *time-edge*. The set of all edges active at any given time $t \in \mathbb{N}$ is denoted $E_t(G, \tau) := \{e \in E(G) : t \in \tau(e)\}$. The latest time Λ for which $E_\Lambda(G, \tau)$ is non-empty is called the *lifetime* of a temporal graph (G, τ) (or equivalently $\Lambda := \max_{e \in E(G)} \max \tau(e)$). Similarly to the notation employed for directed graphs, we define the *underlying static graph* of (G, τ) to be the static graph $u(G, \tau) := (V(G), \bigcup_{t \in \mathbb{N}} E_t(G, \tau))$ given by taking every edge that is ever active in (G, τ) to be an edge in G. If every edge of G is active at at-least one time, then $u(G, \tau)$ and G are isomorphic.

There are also other natural restrictions and/or attributes that can be added to the model of a temporal graph that have been studied in the literature. Two that we mention here are temporal graphs with *random* or *periodic* behaviour (we note that these notions have many natural analogues in the more general setting of $\mathbb{T}$-temporal networks as well [22]).

Just as in the static setting, random behaviour can be very useful for modelling temporal graphs under imperfect information [79]. Although we will not consider random temporal graphs in this thesis, we point out that, similarly to the static setting, exploiting randomness has been effective tool for the theoretical study of

both structural and algorithmic problems on temporal graphs (see, the surveys by Michail [79] or Casteigts et. al. [22] for more information).

It turns out that periodicity is also a very natural restriction on temporal structure since many real-world-networks display periodic recurrences of edge-appearences and disappearences. Some well-known examples of periodic temporal graphs are the public-transportation and other logistical networks [22,79] or low Earth-orbiting satellite systems [22]. There are different formalisations of what a periodic temporal graph should be (for example there might be a global period which governs edge-re-appearance or perhaps each edge might have its own private period). We will not study periodic temporal graphs of this kind; however we point out that in Section 4.4 we will show that knowledge of some global (i.e. over all edges) upper- and lower-bounds on successive edge-appearances in temporal graphs can be used to strengthen the algorithmic results that we will prove in Section 4.3.

4.1.2 The graph theory and complexity theory of temporal graphs

Clearly, any graph theoretic notion that is familiar from the setting of static-graphs can be naïvely applied to temporal graphs by simply defining it in terms of the underlying static graph. Although this can be sometimes useful, it is in general too restrictive since it forgets the temporal structure completely. In fact one is often most interested in truly temporal analogues of standard graph theoretic notions (e.g. connectivity or distance) that take temporal information into account.

To illustrate this point, notice that there are two natural notions of walk in a temporal graph: one is the familiar notion of a walk in static graphs and the other is a truly temporal notion where we require consecutive edges in walks to appear at non-decreasing times. Formally, given vertices x and y in a temporal graph (G,τ), a *temporal* (x,y)*-walk* is a sequence $W=(e_1,t_1),\dots,(e_n,t_n)$ of time-edges such that $e_1,\dots,e_n$ is a walk in G starting at x and ending at y and such that $t_1\leq t_2\leq\cdots\leq t_n$. If $n>1$, we denote by $W-(e_n,t_n)$ the temporal walk $(e_1,t_1),\dots,(e_{n-1},t_{n-1})$. We call a temporal (x,y)-walk *closed* if $x=y$ and we call it a *strict temporal walk* if the times of the walk form a strictly increasing sequence. Consider for example the temporal graph in Figure 4.1. In this graph, the sequence $(ab,1)(bc,10)(ca,101)$ of time-edges constitutes a temporal cycle in (K_3,τ). Temporal graphs naturally induce permissible edge-orderings in walks; for example notice that all temporal cycles in the temporal graph in Figure 4.1 must first visit ab before any other edge.

Two examples of applications in which it might be appropriate to prefer strict or non-strict temporal

walks respectively are train networks or certain epidemiological networks. In the first case, it does not make sense to consider non-strict walks since we cannot take multiple consecutive train journeys that all depart at the same time. In contrast, if, in the context of disease transmission, the presence of a time-edge indicates that two people are close enough for contagion at that time, then non-strict walks may make more sense (since a connected set of edges all present at the same time might indicate that all of those people are very close to each-other). Hereafter we will assume all temporal walks to be strict.

Many other standard graph-theoretic notions such as degree, distance and connectivity (to name a few) have (sometimes many different) temporal counterparts. In some cases defining such a temporal counterpart is straightforward: consider for example the degree of a vertex v in a temporal graph (G, τ). Since this might vary at different time-points, it is natural to distinguish between the *underlying degree* (or simply *degree*) $d(v) := d_{u(G,\tau)}(v)$ of any vertex v, the *t-degree* (or *degree at time t*) $d^t(v) := d_{(V(G),E_t(G,\tau))}(v)$ of v and the *temporal total degree* $d^{tot}(v) := \sum_{t\in\mathbb{N}} d^t(v)$ (which counts the number of time-edges incident with v).

In other cases, it is not as straightforward to determine which temporal analogue is the right one. To illustrate this point, suppose we wished to define a temporal analogue of distance between two vertices. Should we define the temporal distance between two vertices u and v to be their distance in the underlying static graph? Or would it be more meaningful for the distance from u to v to be the length of the shortest temporal path? Or, then again, should the temporal distance between u and v perhaps represent the minimum possible time that it takes to travel from u to v via a temporal walk? All of these notions (among others still) have been considered before and we refer the interested reader to the survey article by Casteigts, Flocchini, Quattrocchi and Santoro [22] where these topics are discussed at length.

Given these considerations, it should not be a surprise to discover that the temporal analogues of standard graph-theoretic notions can behave quite differently from their static counterparts. For example, consider the question of finding the strongly-connected-components in directed graphs. This is a prototypical example of a polynomial-time problem for static graphs, but its temporal analogue (where two vertices are strongly connected if there are temporal paths connecting them in both directions) was shown to be NP-hard by Bhadra and Ferreira [14].

Another striking example of this phenomenon is Kempe, Kleinberg and Kumar's result which shows that it is NP-hard to determine the maximum-number of internally-vertex-disjoint temporal (s,t)-paths in temporal graphs [66] (another problem that is well-known to be tractable in the static setting). In particular, the naïve

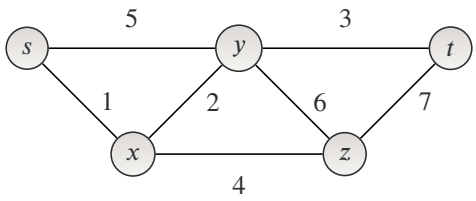


Figure 4.2: A counterexample to Menger's theorem for temporal graphs taken from Michail's survey article [79] which was in-turn adapted from a paper of Kempe, Klienberg and Kumar [66]. Notice that, although there are no two internally-vertex-disjoint temporal paths from s to t, after we remove any one of x, y or z, the vertex s can still reach t via a temporal walk.

temporal analogue of Menger's theorem on connectivity fails for temporal graphs. Recall that Menger's theorem states that, given any two distinct vertices s and t in any graph G, the minimum number of vertices that we need to remove in order to separate s from t in G is equal to the maximum number of internally-vertex-disjoint (s,t)-paths in G (see Diestel's textbook for a proof [30, Theorem 3.3.1]). In contrast, there exist temporal graphs in which, given two distinct vertices s and t, the number of vertices that we need to remove in order to destroy all temporal (s,t)-walks is strictly greater than the maximum number of internally-vertex-disjoint temporal (s,t)-walks (see Figure 4.2 for an illustration).

There are sometimes instances of graph-theoretic notions that generalize nicely to the temporal setting. One example of this phenomenon is the *edge*-version of Menger's theorem: Berman showed that the minimum number of edges that one needs to remove in order to destroy all temporal (s,t)-walks (for any two vertices s and t) is equal to the maximum number of edge-disjoint temporal (s,t)-walks [10]. Furthermore, by considering a different notion of distjointness of paths (roughly one might think of it in terms of departure time disjointness) a vertex-centric temporal analogue of Menger's theorem can indeed be recovered [76].

In general, from a complexity-theoretic perspective, decision problems tend to get significantly harder on temporal graphs than they are on static graphs [14, 66, 79]. As we already mentioned, the natural temporal variants of many polynomial-time solvable problems on static graphs become NP-hard. Some examples are: finding connected components [14], determining the maximum number of internally vertex-disjoint (s,t)-paths [66], finding maximum matchings [78] or deciding if a graph is temporally Eulerian [18, 75]. Generally speaking (as one would expect) passing from static to temporal graphs does not make NP-hard problems any easier: for example Michail and Spirakis showed that various temporal analogues of the Travelling Salesman Problem as well as the Path-packing, Max-TSP and Minimum-Cycle-Cover problems are all NP-hard [79, 80].

Thus, given all of the evidence for computational intractability in the temporal setting, it is natural to seek structural restrictions on the input that yield islands of tractability. To this end, Enright, Meeks, Mertzios and

Zamaraev showed that parameterizations by underlying tree-width (i.e. the tree-width of the underlying static graph) together with other parameters can yield tractability results in some cases [37]. Unfortunately, restricting the underlying static structure is often not enough since there are many decision problems that remain NP-hard even in the extremely restrictive cases in which the underlying static graph of the input is a cactus graph (as we shall prove in Section 4.2) or a tree [78] or even a star [3]! In fact, these considerations rule out fixed-parameter tractability results parameterized by other temporal variants of structural measures on static graphs as well. Examples of such measures are temporal variants of feedback vertex number [23] or other tree-width analogues such as *layer-* or *slice-tree-width* [44].

4.1.3 Chapter overview

Having introduced some background on temporal graphs, we will now conclude the introduction to this chapter by giving a more specific contextualization as well as a high-level overview of the technical contributions of this chapter.

Some of the most natural and most studied topics in the theory of temporal graphs are temporal walks, paths and corresponding notions of temporal reachability [2, 6, 14, 23, 66, 76, 95, 96]. Related to these notions is the study of explorability of a temporal graph which asks whether it is possible to visit all vertices or edges of a temporal graph via some temporal walk.

Temporal vertex-exploration problems (such as temporal variants of the Travelling Salesman problem) have already been thoroughly studied [3, 40, 80]. In contrast, the rest of this chapter will focus on temporal *edge*-exploration and specifically we study *temporally Eulerian graphs*. Informally, these are temporal graphs admitting a temporal circuit that visits every edge at exactly one time (i.e. a temporal circuit that yields an Euler circuit in the underlying static graph).

Deciding whether a static graph is Eulerian is a prototypical example of a polynomial time solvable problem. In fact this follows from Euler's characterization of Eulerian graphs dating back to the 18^{th} century [41]. In contrast, we will show in Section 4.2 that, unless P = NP, a characterization of this kind cannot exist for temporal graphs. In particular we show that deciding whether a temporal graph is *temporally Eulerian* is NP-complete even if strong restrictions are placed on the structure of the underlying graph and each edge is active at only three times.

As we already mentioned in Section 4.1.2, the existence of problems that are tractable on static graphs, but

NP-complete on temporal graphs is well-known [3,21,77,79]. Furthermore, since there are examples of problems whose temporal analogues remain hard even on trees [3,77], the known parameters (such as the temporal variants of feedback vertex number [23] and tree-width [44] which were mentioned in the previous section) will be of no use to us here since the problems we consider remain NP-complete even when these measures are bounded by constants on the underlying static graph. To overcome these difficulties, in Section 4.3 we introduce a new purely-temporal parameter called *interval-membership-width*. Parameterizing by this measure we find that the problem of determining whether a temporal graph is temporally Eulerian is in FPT.

Temporal graphs of low interval-membership-width are 'temporally sparse' in the sense that only few edges are allowed to appear both before and after any given time. We point out that this parameter does *not* depend on the structure of the underlying static graph, but it is instead influenced only by the temporal structure. We believe that interval-membership-width will be a parameter of independent interest for other temporal graph problems in the future.

In Section 4.4 we will show that our study of temporally Eulerian graphs is closely related to a temporal variant of the Travelling Salesman Problem concerning the exploration of temporal stars via a temporal circuit which starts at the center of the star and which visits all leaves. Akrida, Mertzios and Spirakis introduced this problem and proved it to be NP-complete on temporal stars in which every edge has at most k appearances times for all $k \geq 6$ [3]. Although they also showed that the problem is polynomial-time solvable whenever each edge of the input temporal star has at most 3 appearances, they left open the question of determining the hardness of the problem when each edge has at most 4 or 5 appearances. We resolve this open problem in the course of proving our results about temporally Eulerian graphs. Combined with Akrida, Mertzios and Spirakis' results, this gives a complete dichotomy: their temporal star-exploration problem is in P if each edge has at most 3 appearances and is NP-complete otherwise.

As a potential 'island of tractability', Akrida, Mertzios and Spirakis proposed to restrict the input to their temporal star-exploration problem by requiring consecutive appearances of the edges to be evenly spaced (by some globally defined spacing). Using our new notion of interval-membership-width we are able to show in Section 4.4 that this restriction does indeed yield tractability parameterized by the maximum number of times per edge (thus partially resolving their open problem). Furthermore, we show that a slightly weaker result also holds for the problem of determining whether a temporal graph is temporally Eulerian in the setting with evenly-spaced edge-times.

4.2 Hardness of temporal edge exploration

Recall that an Euler circuit in a static graph G is a circuit $e_1 \dots, e_m$ which traverses every edge of G exactly once. In this section we are interested in the natural temporal analogue of this notion. We point out that, independently and simultaneously to our work here, Marino and Silva also studied temporal variants of the problem of deciding whether a graph is Eulerian [75].

Definition 4.2.1. A *temporal Eulerian circuit* in a temporal graph (G, τ) is a closed temporal walk $(e_1, t_1), \dots, (e_m, t_m)$ such that $e_1 \dots, e_m$ is an Euler circuit in the underlying static graph G. If there exists a temporal Eulerian circuit in (G, τ), then we call (G, τ) *temporally Eulerian*.

Note that if (G, τ) is a temporal graph in which every edge appears at exactly one time, then we can determine whether (G, τ) is temporally Eulerian in time linear in $|E(G)|$. To see this, note that, since every edge is active at precisely one time, there is only one candidate ordering of the edges (which may or may not give rise to an Eulerian circuit). Thus it is clear that the number of times per edge is relevant to the complexity of the associated decision problem – which we state as follows.

TEMPEULER(k)

Input: A temporal graph (G, τ) where $|\tau(e)| \leq k$ for every edge e in the graph G.

Question: Is (G, τ) *temporally Eulerian*?

As we mentioned in Section 4.1, here we will show that TEMPEULER(k) is related to an analogue of the Travelling Salesman problem on temporal stars [3]. This problem (denoted as STAREXP(k)) was introduced by Akrida, Mertzios and Spirakis [3]. It asks whether a given temporal star (S_n, τ) (where S_n denotes the n-leaf star) with at most k times on each edge admits a closed temporal walk starting at the center of the star and which visits every leaf of S_n. We call such a walk an *exploration* of (S_n, τ). A temporal star that admits an exploration is called *explorable*. Formally we have the following decision problem.

STAREXP(k)

Input: A temporal star (S_n, τ) where $|\tau(e)| \leq k$ for every edge e in the star S_n.

Question: Is (S_n, τ) explorable?

We will now show that TEMPEULER(k) is NP-complete for all k at least 3 (Corollary 4.2.5) and that STAREXP(k) is NP-complete for all k at least 4 (Corollary 4.2.3). This last result resolves an open problem of Akrida, Mertzios and Spirakis which asked to determine the complexity of STAREXP(4) and STAREXP(5) [3].

To show that STAREXP(4) is NP-hard, we will provide a reduction from the 3-COLORING problem (see for instance Garey and Johnson [48] for a proof of NP-completeness) which asks whether an input graph G is 3-colorable.

3-COLORING

Input: A finite simple graph G.

Question: Does G admit a proper 3-coloring?

Throughout, for an edge e of a temporal star (S_n, τ), we call any pair of times $(t_1, t_2) \in \tau(e)^2$ with $t_1 < t_2$ a *visit of* e. We say that e is *visited at* (t_1, t_2) in a temporal walk if the walk proceeds from the center of the star along e at time t_1 and then back to the center at time t_2. We say that two visits (x_1, x_2) and (y_1, y_2) of two edges e_x and e_y are *in conflict* with one another (or that 'there is a conflict between them') if there exists some time t with $x_1 \leq t \leq x_2$ and $y_1 \leq t \leq y_2$. Note that a complete set of visits (one visit for each edge of the star) which has no pairwise conflicts is in fact an exploration.

Theorem 4.2.2. STAREXP(4) *is* NP*-hard.*

Proof. Take any 3-COLORING instance G with vertices $\{x_1, \ldots, x_n\}$. We will construct a STAREXP(4) instance (S_p, τ) (where $p = n + 3m$) from G.

Defining S_p. The star S_p is defined as follows: for each vertex x_i in G, we make one edge e_i in S_p while, for each edge $x_i x_j$ with $i < j$ in G, we make three edges e^0_{ij}, e^1_{ij} and e^2_{ij} in S_p.

Defining τ. For $i \in [n]$ and any non-negative integer $\psi \in \{0, 1, 2, \ldots\}$, let t^i_ξ be the integer

$$t^i_\psi := 2in^2 + 2\psi(n+1) \tag{4.1}$$

and take any edge $x_j x_k$ in G with $j < k$. Using the times defined in Equation (4.1) and taking $\xi \in \{0, 1, 2\}$, we

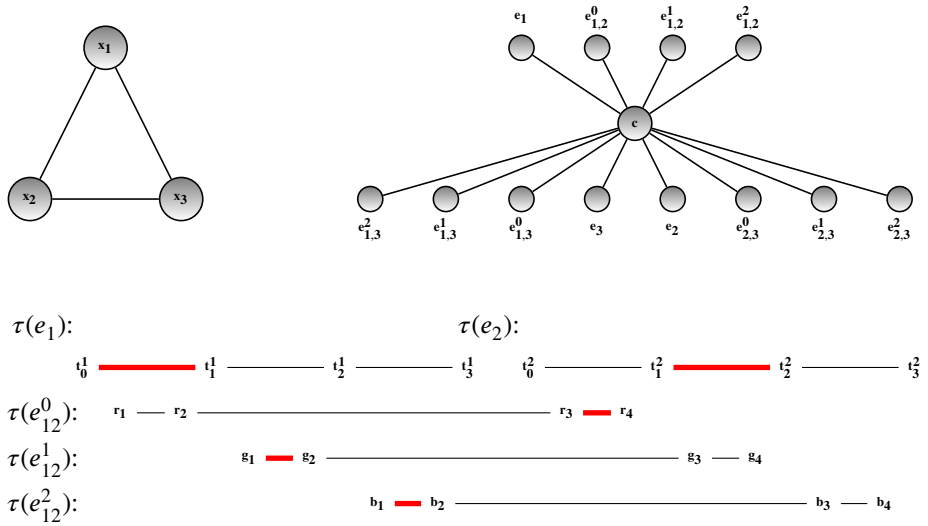


Figure 4.3: Top left: K^3; we assume the coloring $x_i \mapsto i-1$. Top right: star constructed from K^3. Bottom: times (and corresponding intervals) associated with the edges e_1, e_2 and $e^0_{1,2}$, $e^1_{1,2}$, $e^2_{1,2}$ (time progresses left-to-right and intervals are not drawn to scale). We write r_1, r_2, r_3, r_4 as shorthand for the entries of $\tau(e^0_{1,2})$ (similarly, for $i \in [4]$, we write g_i and b_i with respect to $\tau(e^1_{1,2})$ and $\tau(e^2_{1,2})$). The red and thick intervals correspond to visits defined by the coloring of the K^3.

then define $\tau(e_i)$ and $\tau(e^{\xi}_{jk})$ as

$$\tau(e_i) := \left\{t^i_0, t^i_1, t^i_2, t^i_3\right\} \text{ and} \tag{4.2}$$

$$\tau(e^{\xi}_{jk}) := \left\{t^j_{\xi} + 2k - 1, \quad t^j_{\xi} + 2k, \quad t^k_{\xi} + 2j - 1, \quad t^k_{\xi} + 2j\right\}. \tag{4.3}$$

Note that the elements of these sets are written in increasing order (see Figure 4.3).

Intuitively, the times associated to each edge $e_i \in E(S_p)$ corresponding to a vertex $x_i \in V(G)$ (Equation (4.2)) encode the possible colorings of x_i via the three possible starting times of a visit of e_i. The three edges e^0_{ij}, e^1_{ij} and e^2_{ij} corresponding to some $x_i x_j \in E(G)$ are instead used to 'force the colorings to be proper' in G. That is to say that, for a color $\xi \in \{0,1,2\}$, the times associated with the edge e^{ξ}_{ij} (Equation (4.3)) will prohibit us from entering e_i at its ξ-th appearance and also entering e_j at its ξ-th appearance (i.e. 'coloring x_i and x_j the same color').

Observe that the first two times in $\tau(e^{\xi}_{jk})$ lie within an interval given by consecutive times in $\tau(e_j)$ and that the same holds for the last two times in $\tau(e^{\xi}_{jk})$ with respect to $\tau(e_k)$ (see Figure 4.3). More precisely, it is immediate that for $1 \leq j < k \leq n$ and $\xi \in \{0,1,2\}$, we have:

$$t^j_{\xi} \; < \; t^j_{\xi} + 2k - 1 < \; t^j_{\xi} + 2k < \; t^j_{\xi+1} \tag{4.4}$$

Given this set-up, we will now show that G is a yes instance if and only if (S_p, τ) is.

Suppose (S_p, τ) is explorable. Define the coloring (to be shown proper) $c : V(G) \to \{0,1,2\}$ taking each vertex x_i to the color ξ whenever e_i is entered at time t^i_ξ within the exploration of (S_p, τ) (note that these are the only possible times at which e_i can be entered, since every edge appears at exactly 4 times). We claim that c is a proper coloring. To see this, suppose on the contrary that there is a monochromatic edge $x_i x_j$ with $i < j$ of color ξ in G. Then, this means that e_i was entered at time t^i_ξ and exited at time at least $t^i_{\xi+1}$ and similarly e_j was entered at time t^j_ξ and exited at time at least $t^j_{\xi+1}$. But then, since all times in $\tau(e^\xi_{ij})$ are contained either in the open interval $(t^i_\xi, t^i_{\xi+1}) \subseteq \mathbb{R}$ or the open interval $(t^j_\xi, t^j_{\xi+1}) \subseteq \mathbb{R}$ we know that e^ξ_{ij} cannot be explored (by (4.4)). This contradicts the assumption that (S_p, τ) is explorable, hence c must be a proper coloring.

Conversely, suppose G admits a proper 3-coloring $c : V(G) \to \{0,1,2\}$. We define the following exploration of (S_p, τ) (see Figure 4.3):

- for every vertex x_i in G, if $c(x_i) = \xi$, then visit e_i at $(t^i_\xi, t^i_{\xi+1})$
- for every edge $x_i x_j$ in G with $i < j$ and every color $\xi \in \{0,1,2\}$, define the visit of e^ξ_{ij} as follows: if $c(x_i) \neq \xi$, then visit e^ξ_{ij} at $(t^i_\xi + 2j - 1, t^i_\xi + 2j)$; otherwise visit e^ξ_{ij} at $(t^j_\xi + 2i - 1, t^j_\xi + 2i)$.

Our aim now is to show that the visits we have just defined in terms of the coloring c are disjoint (and thus witness the explorability of (S_p, τ)).

Take any $i < j$ and any $\xi \in \{0,1,2\}$. By our definition of $\tau(e_i)$ and $\tau(e_j)$, we must have $\max \tau(e_i) = t^i_3 < 2jn^2 = \min \tau(e_j)$ whenever $i < j$. Thus we note that there are no conflicts between the visit of e_i and the visit of e_j.

Note that, for all $(\xi, \omega) \in \{0,1,2\}^2$ and all pairs of edges $x_i x_j$ and $x_k x_\ell$ in G with $i < j$ and $k < \ell$, the visit $(v_{i,j}, v_{i,j} + 1)$ of e^ξ_{ij} is in conflict with the visit $(v_{k,\ell}, v_{k,\ell} + 1)$ of $e^\omega_{k\ell}$ only if $x_i x_j = x_k x_\ell$. To see this, observe that the visits of e^ξ_{ij} and $e^\omega_{k\ell}$ both consist of two consecutive times where the first time is odd. Thus we would only have a conflict if $v_{i,j} = v_{k,\ell}$ which can be easily checked to happen only if $i = k$ and $j = \ell$.

Finally we claim that there are no conflicts between the visit of e^ξ_{ij} and the visits of either e_i or e_j. To show this, we will only argue for the lack of conflicts between the visits of e_i and e^ξ_{ij} since the same ideas suffice for the e_j-case as well. Suppose $c(x_i) = \xi$, then we visit e^ξ_{ij} at $(t^j_\xi + 2i - 1, t^j_\xi + 2i)$ and then $t^j_\xi + 2i - 1 > t^j_\xi > t^i_3 = \max \tau(e_i)$ since $i < j$ and since c is a proper coloring. Similarly, if $c(x_i) \neq \xi$, then we visit e^ξ_{ij} at $(t^i_\xi + 2j - 1, t^i_\xi + 2j)$. As

we observed in Inequality (4.4), we have $t^i_\xi < t^i_\xi + 2j - 1 < t^i_\xi + 2j < t^i_{\xi+1}$. Thus, if (u_1, u_2) is the visit of e_i, then either $u_2 < t^i_\xi$ or $t^i_{\xi+1} < u_1$. In other words, no conflicts arise.

This concludes the proof since we have shown that the visits we assigned to the edges of S_p constitute an exploration of (S_p, τ). ■

Observe that increasing the maximum number of times per edge cannot make the problem easier: we can easily extend the hardness result to any $k' > 4$ by simply adding a new edge with k' times all prior to the times that are already in the star. This, together with the fact that Akrida, Mertzios and Spirakis [3] showed that StarExp(k) is in NP for all $k \geq 0$, allows us to conclude the following corollary.

Corollary 4.2.3. *For all k at least 4,* StarExp(k) *is* NP*-complete.*

Next we shall reduce StarExp(k) to TempEuler($k-1$). We point out that, for our purposes within this section, only the first point of the statement of the following result is needed. However, later (in the proof of Corollary 4.3.4) we shall make use of the properties stated in the second point of Lemma 4.2.4 (this is also why we allow any k times per edge rather than just considering the case $k = 4$). Thus we include full details here.

Lemma 4.2.4. *For all $k \geq 2$ there is a polynomial-time-computable mapping taking every* StarExp(k) *instance (S_n, τ) to a* TempEuler($k-1$) *instance (D_n, σ) such that*

1. *(S_n, τ) is a yes instance for* StarExp(k) *if and only if (D_n, σ) is a yes instance for* TempEuler($k-1$) *and*
2. *D_n is a graph obtained by identifying n-copies $\{K^3_1, \ldots, K^3_n\}$ of a cycle on three vertices along one center vertex (see Figure 4.4) and such that*

$$\max_{t \in \mathbb{N}} |\{e \in E(D_n) : \min(\sigma(e)) \leq t \leq \max(\sigma(e))\}|$$
$$\leq 3 \max_{t \in \mathbb{N}} |\{e \in E(S_n) : \min(\tau(e)) \leq t \leq \max(\tau(e))\}|.$$

Proof. Note that we can assume without loss of generality that: (1) every edge in S_n has exactly k-times on each edge and (2) that all times are multiples of 2. This follows from the fact that we can construct from any StarExp(k)-instance $(S_{n'}, u)$ another StarExp(k)-instance (S_n, τ) so that : (S_n, τ) is explorable if and only if $(S_{n'}, u)$ also is and every time in (S_n, τ) satisfies conditions (1) and (2).

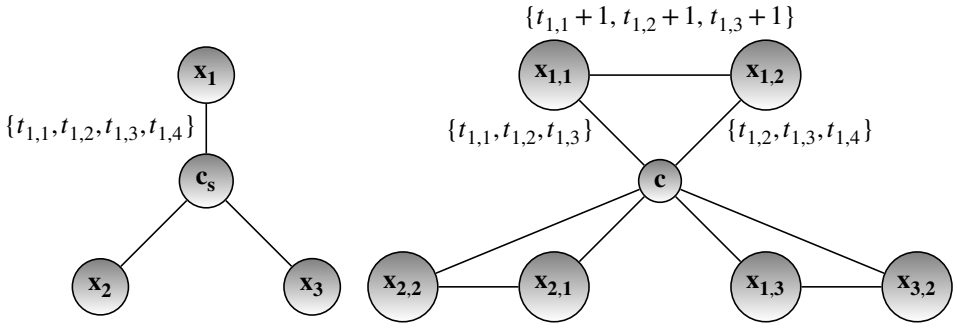


Figure 4.4: Building (D_3, σ) from (S_3, τ). The times along edges are drawn only for the edge $c_s x_1$ in S_3 and for its corresponding 3-cycle $c x_{1,1} x_{1,2}$ in D_3. Since $t_{1,1}$, $t_{1,2}$, $t_{1,3}$ and $t_{1,4}$ are all multiples of 2, we know that $t_{1,j} < t_{1,j+1} < t_{1,j+1}$ for all $j \in [3]$. Thus the reduction associates the visit (t_s, t_e) of $c_s x_1$ in the star to exploration $(t_s, t_s + 1, t_e)$ of the 3-cycle corresponding to $c_s x_1$ in D_3.

Now we will show how to construct a TEMPEULER$(k-1)$-instance (D_n, σ) from (S_n, τ) such that (D_n, σ) is temporally Eulerian if and only if (S_n, τ) is explorable (see Figure 4.4). Throughout, denote the vertices of the i-th 3-cycle C_i^3 of D_n by $\{c, x_{i,1}, x_{i,2}\}$ and let its edges be $f_{i,1} = c x_{i,1}$, $f_{i,2} = x_{i,1} x_{i,2}$ and $f_{i,3} = x_{i,2} c$. For every $i \in [n]$ with $\tau(e_i) = \{t_1, \ldots, t_k\}$ where $t_1 < t_2 < \cdots < t_k$, define the map $\sigma : E(D_n) \to 2^{\mathbb{N}}$ as:

$$\sigma(f_{i,1}) := \{t_1, \ldots, t_{k-1}\},$$

$$\sigma(f_{i,2}) := \{t_1 + 1, \ldots, t_{k-1} + 1\},$$

$$\sigma(f_{i,3}) := \{t_2, \ldots, t_k\}.$$

Note that $|\sigma(f_{i,1})| = |\sigma(f_{i,2})| = |\sigma(f_{i,3})| = k - 1$. Now suppose (S_n, τ) is a yes-instance witnessed by the sequence $\mathcal{V}$ of visits $\mathcal{V} := (x_1, y_1), \ldots, (x_n, y_n)$ of the edges $e_1, \ldots, e_n$ of S_n and observe that $y_i < x_{i+1}$ for all $i \in [n-1]$. We claim that the sequence of time-edges

$$(f_{1,1}, x_1), (f_{1,2}, x_1 + 1), (f_{1,3}, y_1), \ldots, (f_{n,1}, x_n), (f_{n,2}, x_n + 1), (f_{n,3}, y_n)$$

is a temporal Eulerian circuit in G. To see this, recall that $y_j < x_{j+1}$ (for $j \in [n-1]$) and note that:

1. by definition $f_{1,1}, f_{1,2}, f_{1,3}, \ldots, f_{n,1}, f_{n,2}, f_{n,3}$ is an Eulerian circuit in the underlying static graph D_n (i.e. we walk along each 3-cycle in turn) and
2. $x_i < x_i + 1 < y_i$ for all $i \in [n]$ since we assumed that $x_i, y_i \in 2\mathbb{N}$.

Conversely, suppose (D_n, σ) is a yes-instance and let this fact be witnessed by the temporal Eulerian circuit $\mathcal{K}$. Recall that a temporal Eulerian circuit induces an Eulerian circuit in the underlying static graph. Thus, since

every Eulerian circuit in D_n must run through each 3-cycle, we know that $\mathcal{K}$ must consist – up to relabelling of the edges – of a sequence of time-edges of the form

$$\mathcal{K} := (f_{1,1}, x_{1,1}), (f_{1,2}, x_{1,2}), (f_{1,3}, y_{1,3}), \ (f_{2,1}, x_{2,1}), (f_{2,2}, x_{2,2}), (f_{2,3}, y_{2,3}), \\ \ldots, \ (f_{n,1}, x_{n,1}), (f_{n,2}, x_{n,2}), (f_{n,3}, y_{n,3}).$$

It follows immediately from the definition of (D_n, σ) that visiting each edge e_j in S_n at $(x_{j,1}, y_{j,3})$ constitutes an exploration of (S_n, τ), as desired. ■

Since TEMPEULER(k) is clearly in NP (where the circuit acts as a certificate), our desired NP-completeness result follows immediately from Lemma 4.2.4 and Corollary 4.2.3.

Corollary 4.2.5. TEMPEULER(k) *is* NP*-complete for all* k *at least* 3.

As we noted earlier, TEMPEULER(1) is trivially solvable in time linear in the number of edges of the underlying static graph. Although our proof leaves open the $k = 2$ case, Marino and Silva closed this gap showing that TEMPEULER(k) is NP-complete for all $k \geq 2$ (thus resolving an open problem from a paper by the author of this thesis and Meeks [18]).

Observe that the reduction in Lemma 4.2.4 rules out FPT algorithms with respect to many standard parameters describing the structure of the underlying graph (for instance the path-width is 2 and feedback vertex number[2] is 1). In fact we can strengthen these intractability results even further by showing that TEMPEULER(k) is hard even for instances whose underlying static graph has vertex-cover number[3] 2. This motivates our search in Section 4.3 for parameters that describe the structure of the times assigned to edges rather than just the underlying static structure.

Notice that this time we will reduce from STAREXP(k) to TEMPEULER(k) (rather than from STAREXP($k+1$) as in Lemma 4.2.4), so, in contrast to our previous reduction (Lemma 4.2.4), the proof of the following result cannot be used to show hardness of TEMPEULER(3).

Theorem 4.2.6. *For all* $k \geq 4$*, the* TEMPEULER(k) *problem is* NP*-complete even on temporal graphs whose underlying static graph has vertex-cover number* 2*.*

[2]Recall that a *feedback vertex set* in a graph (resp. directed graph) G is a vertex subset $S \subseteq V(G)$ such that $G - S$ is a forest (resp. directed acyclic graph). The *feedback vertex number* of a graph G is the minimum number of vertices needed to form a feedback vertex set in G.

[3]Recall that a *vertex-cover* in a graph G is a vertex-subset $S \subseteq V(G)$ such that every edge in G is incident with at least one vertex in S. The *vertex-cover number* of a graph G is the minimum number of vertices needed to form a vertex-cover of G.

Proof. Take any StarExp(k) instance (S_n, τ) and assume that n is even (if not, then simply add a dummy edge with all appearances strictly after the lifetime of the graph). Denoting by c the center of S_n and by $x_1, \ldots, x_n$ its leaves, let $S_n^{c_1,c_2}$ be the double star constructed from S_n by splitting c into two twin centers; stating this formally, we define $S_n^{c_1,c_2}$ as

$$S_n^{c_1,c_2} = (\{c_1, c_2, x_1, \ldots, x_n\}, \{c_i x_j : i \in [2] \text{ and } j \in [n]\}).$$

Notice that, since n is even, $S_n^{c_1,c_2}$ is Eulerian and notice that the set $\{c_1, c_2\}$ is a vertex cover of $S_n^{c_1,c_2}$.

Defining $\sigma : E(S_n^{c_1,c_2}) \to 2^{\mathbb{N}}$ for all $i \in [2]$ and $j \in [n]$ as $\sigma(c_i x_j) = \tau(c x_j)$, we claim that the temporal graph $(S_n^{c_1,c_2}, \sigma)$ is temporally Eulerian if and only if (S_n, τ) is explorable.

Suppose that (S_n, τ) is explorable and let this be witnessed by the sequence of visits $(s_1, t_1), \ldots, (s_n, t_n)$. Then it follows immediately by the definition of σ that the following sequence of time-edges is a temporal circuit in $(S_n^{c_1,c_2}, \sigma)$:

$$(c_1 x_1, s_1), (x_1 c_2, t_1), (c_2 x_2, s_2)(x_2 c_1, t_2), (c_1 x_3, s_3), (x_3 c_2, t_3), \ldots, (c_2 x_n, s_n)(x_n c_1, t_n).$$

To see this, note that this clearly induces an Eulerian circuit in the underlying static graph $S_n^{c_1,c_2}$; furthermore, since $(s_1, t_1), \ldots, (s_n, t_n)$ is an exploration in (S_n, τ), it follows that $s_1 < t_1 < s_2 < t_2 < \ldots < s_n < t_n$, as desired.

Suppose now that $(S_n^{c_1,c_2}, \sigma)$ is temporally Eulerian and that this fact is witnessed (without loss of generality – up to relabeling of vertices) by the temporal Eulerian circuit

$$(c_1 x_1, s_1), (x_1 c_2, t_1), (c_2 x_2, s_2)(x_2 c_1, t_2), (c_1 x_3, s_3), (x_3 c_2, t_3), \ldots, (c_2 x_n, s_n)(x_n c_1, t_n).$$

Then, by the definition of σ in terms of τ and by similar arguments to the previous case, we have that $(s_1, t_1), \ldots, (s_n, t_n)$ is an exploration of (S_n, τ). ■

4.3 Interval-membership-width

As we saw in the previous section, both TempEuler(k) and StarExp($k+1$) are NP-complete for all $k \geq 3$ even on instances whose underlying static graphs are very sparse (for instance even on graphs with vertex cover number 2). Clearly this means that any useful parameterization must take into account the *temporal*

structure of the input. As we discussed previously, other authors have already proposed measures of this kind such as the temporal feedback vertex number [23] or temporal analogues of tree-width [44]. However these measures are all bounded on temporal graphs for which the underlying static graph has bounded feedback vertex number and tree-width respectively. Our reductions therefore show that TEMPEULER(k) is para-NP-complete with respect to these parameters. Thus we do indeed need some new measure of temporal structure. To that end, here we introduce such a parameter called *interval-membership-width* which depends only on temporal structure and not on the structure of the underlying static graph. Parameterizing by this measure, we will show that both TEMPEULER(k) and STAREXP(k) lie in FPT.

To first convey the intuition behind our width measure, consider again the TEMPEULER(1) problem. As we noted earlier, this is trivially solvable in time linear in $|E(G)|$. The same is true for any TEMPEULER(k)-instance (G, τ) in which every edge is assigned a 'private' interval of times: that is to say that, for all distinct edges e and f in G, either $\max \tau(f) < \min \tau(e)$ or $\max \tau(e) < \min \tau(f)$. This holds because, on instances of this kind, there is only one possible relative ordering of edges available for an edge-exploration. It is thus natural to expect that, for graphs whose edges have intervals that are 'almost private' (defined formally below), we should be able to deduce similar tractability results.

Towards a formalization of this intuition, suppose that we are given a temporal graph (G, τ) which has precisely two edges e and f such that there is a time t with $\min \tau(e) \leq t \leq \max \tau(e)$ and $\min \tau(f) \leq t \leq \max \tau(f)$. It is easy to see that the TEMPEULER(k) problem is still tractable on graphs such as (G, τ) since there are only two possible relative edge-orderings for an edge exploration of (G, τ) (depending on whether we choose to explore e before f or f before e). These observations lead to the following definition of *interval-membership-width* of a temporal graph (see Figure 4.5).

Definition 4.3.1. The *interval membership sequence* of a temporal graph (G, τ) is the sequence $(F_t)_{t \in [\Lambda]}$ of edge-subsets of G where $F_t := \{e \in E(G) : \min \tau(e) \leq t \leq \max \tau(e)\}$ and Λ is the lifetime of (G, τ). The *interval-membership-width* of (G, τ) is the integer $\mathbf{imw}(G, \tau) := \max_{t \in \mathbb{N}} |F_t|$.

Note that a temporal graph has unit interval-membership-width if and only if every edge is active at times spanning a 'private interval'. Furthermore, we point out that the interval membership sequence of a temporal graph is not the same as the sequence $(E_t(G, \tau))_{t \in \mathbb{N}}$. In fact, although $\max_{t \in \mathbb{N}} |E_t(G, \tau)| \leq \mathbf{imw}(G, \tau)$, there exist classes $\mathcal{C}$ of temporal graphs with unbounded interval-membership-width but such that every temporal graph in $\mathcal{C}$ satisfies the property that at most one edge is active at any given time. To see this consider any

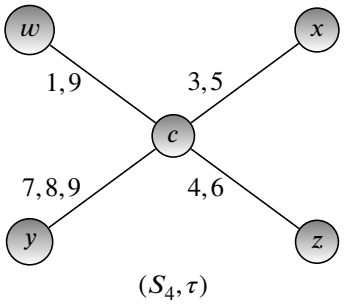


Figure 4.5: A temporal star (S_4, τ) with interval-membership-sequence: $F_1 = F_2 = \{cw\}$, $F_3 = \{cw, cx\}$, $F_4 = F_5 = \{cw, cx, cy\}$, $F_6 = \{cw, cy\}$ and $F_7 = F_8 = F_9 = \{cw, cz\}$.

graph H with edges $e_1, \ldots, e_m$ and let (H, ν) be the temporal graph defined by $\nu(e_i) := \{i, m+i\}$. Clearly $\max_{i \in \mathbb{N}} |E_i(H, \nu)| = 1$, but we have $\mathbf{imw}(H, \nu) = m$.

Note that the interval membership sequence of a temporal graph (G, τ) can be computed in time $\mathcal{O}(\mathbf{imw}(G, \tau) \cdot \Lambda)$ by iterating over the edges of G.

Armed with the notion of interval-membership-width, we will now show that both TEMPEULER(k) and STAREXP(k) are in FPT when parameterized by this measure. We will do so first for TEMPEULER(k) (Theorem 4.3.2) and then we will leverage the reduction of Lemma 4.2.4 to deduce the fixed-parameter-tractability of STAREXP(k) as well (Corollary 4.3.4).

Theorem 4.3.2. *There is an algorithm that decides whether any temporal graph (G, τ) with n vertices and lifetime Λ is a yes-instance of* TEMPEULER(k) *in time $\mathcal{O}(w^3 2^w \Lambda)$ where $w = \mathbf{imw}(G, \tau)$ is the interval-membership-width of (G, τ).*

Proof. Let $(F_t)_{t \in [\Lambda]}$ be the interval membership sequence of (G, τ) and suppose without loss of generality that F_1 is not empty.

We will now describe an algorithm that proceeds by dynamic programming over the sequence $(F_i)_{i \in [\Lambda]}$ to determine whether (G, τ) is temporally Eulerian. For each set F_i we will compute a set $L_i \subseteq F_i^{\{0,1\}} \times V(G) \times V(G)$ consisting of triples of the form (f, s, x) where s and x are vertices in G and f is a function mapping each edge in F_i to an element of $\{0, 1\}$. Intuitively each entry (f, s, x) of L_i corresponds to the existence of a temporal walk starting at s and ending at x at time at most i and such that, for any edge $e \in F_i$, we will have $f(e) = 1$ if and only if e was traversed during this walk.

We will now define the entries L_i recursively starting from the dummy set $L_0 := \{(\mathbf{0}, x, x) : \exists e \in F_1 \text{ incident with } x\}$ where $\mathbf{0} : e \in F_1 \mapsto 0$ is the function mapping every element in F_1 to 0. Take any (f, s, y)

in $F_i^{\{0,1\}} \times V(G) \times V(G)$. For (f,s,y) to be in L_i we will require there to be an entry (g,s,x) of L_{i-1} such that

$$g(e) = 1 \text{ for all } e \in F_{i-1} \setminus F_i \tag{4.5}$$

and such that the one of the following cases holds: either

C1 $y = x$ and $f(e) = 1$ if and only if $e \in F_{i-1} \cap F_i$ and $g(e) = 1$,

or

C2 there exists an edge xy in G such that:

C2.P1 $xy \in E_i(G,\tau) \setminus \{e \in F_i : g(e) = 1\}$ and

C2.P2 $f(e) = 1$ if and only if $g(e) = 1$ or $e = xy$.

The Cases **C1** and **C2** correspond to the the two available choices we have when extending a temporal (s,x)-walk at time i: either we stay put at x (Case **C1**) or we find some new edge xy active at time i (Case **C2**) which has never been used before (Property **C2.P1**) and add it to the walk (Property **C2.P2**). Equation (4.5) ensures that we filter out partial solutions that we already know cannot be extended to a Eulerian circuit. To see this, note that, if an edge e will never appear again after time $i-1$ and we have $g(e) = 0$, then there is no way of extending the temporal walk represented by the triple (g,s,x) to an Eulerian circuit in (G,τ) because one edge will always be left out (namely the edge e).

We claim that the input (G,τ) is temporally Eulerian if and only if L_Λ contains an entry $(\mathbf{1},s,x)$ with $s = x$ and such that $\mathbf{1}$ is the constant all-1 function $\mathbf{1} : y \in F_\Lambda \mapsto 1$. To show this, we will prove the following stronger claim.

Claim 4.3.3. For all $i \in [\Lambda]$, L_i contains an entry (f,s,x) if and only if there exists a temporal walk $(e_1,t_1)\dots(e_p,t_p)$ starting at s and ending at x with $t_p \leq i$ and in which no edge is repeated and such that:

IH1 $(F_1 \cup \dots \cup F_{i-1}) \setminus F_i \subseteq \{e_1,\dots,e_{p-1}\}$ (i.e. every edge whose last appearance is before time i is traversed by the walk) and

IH2 for all $e \in F_i$, we have $f(e) = 1$ if $e \in \{e_1,\dots,e_p\}$ and $f(e) = 0$ otherwise (i.e. f correctly records which edges in F_i have been used in a walk).

Proof of Claim 4.3.3. We show this by induction on i. The Claim holds trivially for $i = 0$, so suppose now that

we are at some time $i > 0$ and hypothesise that the Claim holds for time $i-1$. Furthermore denote by $W_i(f,s,x)$ the set of all temporal temporal (s,x)-walks $(e_1,t_1)\dots(e_p,t_p)$ with $t_p \leq i$ which satisfy Properties **IH1** and **IH2**.

($\Longrightarrow$) First we will show that if (f,s,y) is in L_i, then $W_i(f,s,x)$ is non-empty. By the construction of L_i, we know that, for (f,s,y) to be in L_i, there must have been an element (g,s,x) of L_{i-1} satisfying Equation (4.5) from which we built (f,s,y) according to either Case **C1** or Case **C2**.

Suppose we applied Case **C1** to add (f,s,y) to L_i (i.e. we 'extended' some walk in $W_{i-1}(g,s,x)$ by deciding not to move). Then $x = y$ and we know that $f(e) = 1$ if and only if $g(e) = 1$. Notice that any walk corresponding to (f,s,y) cannot fail to visit some edge in $E_{i-1}(G,\tau)$ that will never again be active after time $i-1$ since we know that (g,s,x) satisfies Equation (4.5). In particular (f,s,y) satisfies Property **IH1** (since g does). Furthermore, since $f(e) = 1$ if and only if $g(e) = 1$ and since g satisfies Property **IH2** (by induction), we know that f must also satisfy Property **IH2**. Thus we have shown that, if we applied Case **C1** to add (f,s,y) to L_i, then $W_i(f,s,y) \neq \emptyset$.

Suppose instead that we applied Case **C2** to add (f,s,y) to L_i. In other words suppose we found an edge xy active at time i with which we wish to extend some walk $W := (e_1,t_1),\dots,(e_p,t_p)$ in $W_{i-1}(g,s,x)$ which starts at s and ends at x. Note that we can infer that $W' := (e_1,t_1)\dots(e_{p-1},t_p)(xy,i)$ is a valid temporal (s,y)-walk with no repeated edges since:

- W has no repeated edges (by the induction hypothesis) and
- xy was not traversed by W (by Property **C2.P1**) and
- $t_p \leq i-1$ (since $W \in W_{i-1}(g,s,x)$).

Thus the fact that g satisfies equation (4.5) combined with the induction hypothesis implies that every edge whose last appearance is before time i is traversed by W' (i.e. W' satisfies Property **IH1**). Furthermore f satisfies Property **IH2** since g does and since $f(e) = 1$ if and only if $g(e) = 1$ or $e = xy$ (by Property **C2.P2**). Thus we have shown that, if $(f,s,y) \in L_i$, then $W_i(f,s,y) \neq \emptyset$.

($\Longleftarrow$) Conversely, we will now show that, if $W_i(f,s,y)$ is non-empty, then $(f,s,y) \in L_i$. Let W' be an element of $W_i(f,s,y)$ and let (xy,j) be the last time-edge traversed by W' (note $j \leq i$).

If $j < i$ then, by the induction hypothesis, there exists an entry $(g,s,y) \in L_{i-1}$ with $W' \in W_{i-1}(g,s,y)$. But then by the construction of L_i from L_{i-1} we have that $(f,s,y) \in L_i$.

Thus suppose $j = i$. Then $W' - (xy, j)$ is a temporal (s, x)-walk ending at time at most $i-1$ satisfying Property **IH1**. Furthermore, by the induction hypothesis, there must be a $(g, s, x) \in L_{i-1}$ which satisfies Equation (4.5) and such that $W' - (xy, j) \in W_{i-1}(g, s, x)$. Now note that, since (f, s, y) satisfies Properties **IH1** and **IH2**, we have that Properties **C2.P1** and **C2.P2** hold as well: thus $(f, s, y) \in L_i$. ■ Claim 4.3.3

Finally we consider the running time of the algorithm. First of all notice that we can compute L_{i+1} from L_i in time at most $(|E_{i+1}(G, \tau)| + 1) \cdot |L_i| \leq (|F_{i+1}| + 1) \cdot |L_i|$. To see this, note that we construct the elements of L_{i+1} by iterating through the elements of L_i and considering for each one the $|F_{i+1}| + 1$ ways of taking a next step in a temporal walk at time i. Since we perform this computation Λ times and since $|F_i| \leq \mathbf{imw}(G, \tau) = w$, the whole algorithm runs in time $\mathcal{O}(w\Lambda \max_{i \in \Lambda} |L_i|)$. Thus all that remains to be shown to complete the proof is that $|L_i|$ is $\mathcal{O}(w^2 2^w)$ for all i. Note that, from its definition, we already know that L_i has cardinality at most $\mathcal{O}(2^w n^2)$ since $L_i \subseteq F_i^{\{0,1\}} \times V(G) \times V(G)$. To improve this bound, we will show that the following two statements hold:

RT1 there exists a time t such that every temporal Eulerian circuit in (G, τ) must start with a vertex incident with an edge in the bag F_t of the interval-membership-sequence of (G, τ);

RT2 for all i, let $\mathcal{X}_i \subseteq V(G)$ be the set of vertices of G defined as $\mathcal{X}_i := \{x \in V(G) : (f, s, x) \in L_i\}$; then $\mathcal{X}_i$ has cardinality at most $4w$, where $w = \mathbf{imw}(G, \tau)$.

To see why it suffices to prove claims **RT1** and **RT2**, notice that they imply that we not only have $L_i \subseteq F_i^{\{0,1\}} \times V(G) \times V(G)$ (which was how we defined L_i in the first place) but in fact there must always exist a $t \in [\Lambda]$ and a subset $\mathcal{X}_i \subseteq V(G)$ (for all i) such that L_i is always of the form

$$L_i \subseteq F_i^{\{0,1\}} \times V(F_t) \times \mathcal{X}_i$$

where both $|V(F_t)|$ and $|\mathcal{X}_i|$ are $\mathcal{O}(w)$. This would clearly then imply that $|L_i|$ is $\mathcal{O}(w^2 2^w)$ for all i, as desired.

*Proof of Claim **RT1**.* Choose $t \in \mathbb{N}$ be greatest possible such that $\bigcup_{j \in [t]} F_j \subseteq F_t$. Suppose by way of contradiction that there exists a temporal Eulerian circuit that starts at a vertex s with s *not* incident with any edge in F_t. Let t' be the earliest time such that the bag $F_{t'}$ contains an edge which which s is incident.

Notice that, since t was chosen greatest possible such that $\bigcup_{j \in [t]} F_j \subseteq F_t$ and since s is not incident with any edge in F_t, it follows that $t' > t$ and that there exists an edge $e \in F_t \setminus F_{t'}$. But then we have a contradiction

since $\max(\tau(e)) \leq t < t'$ and, by time t', e has not yet been visited by the temporal Eulerian circuit starting at s (i.e. any such circuit never vists the edge e). ■ Claim **RT1**

*Proof of Claim **RT2**.* Seeking a contradiction, suppose $|\mathcal{X}_i| \geq 4w+1$. Since $|F_i| \leq w$, the set $\mathcal{X}_{\notin i}$ of elements of $\mathcal{X}_i$ that are not incident with any edge in F_i consists of at least $2w+1$ vertices. Let $\xi : \mathcal{X}_{\notin i} \to E(G)$ be the map associating to each vertex z in $\mathcal{X}_{\notin i}$ an edge $\xi(z)$ incident with z in G such that the last appearance of $\xi(z)$ is latest possible.

Pick a vertex $z \in \mathcal{X}_{\notin i}$ such that $\max \tau(\xi(z)) \leq \max \tau(\xi(z'))$ for any other $z' \in \mathcal{X}_{\notin i}$. Since $\mathcal{X}_{\notin i}$ contained at least $2w+1$ elements and since $|F_{\max \tau(\xi(z))} \cup F_i| \leq 2w$, there must be an element $y \in \mathcal{X}_{\notin i}$ such that

$$\max \tau(\xi(z)) < \min \tau(\xi(y)) < \max \tau(\xi(y)) < i.$$

By the definition of $\mathcal{X}_i$ and since $z \in \mathcal{X}_i$, there is some $(f, s, z) \in L_i$ and, by the previous Claim, there is a walk $W \in \mathcal{W}(f, s, z)$. Notice that, since W ends at the vertex z, it must be that the last time we 'took a step' on W was at a time at most $\max \tau(\xi(z))$; in particular this means that we did not move from z at time i. But then, since $\max \tau(\xi(y)) < i$, y never appears again after time $i-1$ and hence W never traverses $\xi(y)$: this contradicts Property **IH1**. ■ Claim **RT2**

■ Theorem 4.3.2

As a corollary of Theorem 4.3.2, we can leverage the reduction of Lemma 4.2.4 to deduce that StarExp(k) is in FPT parameterized by the interval-membership-width.

Corollary 4.3.4. *There is an algorithm that decides whether a* StarExp(k) *instance* (S_n, τ) *is explorable in time* $\mathcal{O}(w^3 2^{3w} \Lambda)$ *where* $w = \mathbf{imw}(S_n, \tau)$ *and* Λ *is the lifetime of the input.*

Proof. By Lemma 4.2.4, we know that there is a polynomial-time reduction that maps any StarExp(k) instance (S_n, τ) to a TempEuler($k-1$)-instance (D_n, σ) such that

$$\max_t |\{e \in E(D_n) : \min(\sigma(e)) \leq t \leq \max(\sigma(e))\}|$$
$$\leq 3 \max_t |\{e \in E(S_n) : \min(\tau(e)) \leq t \leq \max(\tau(e))\}|.$$

In particular this implies that $\mathbf{imw}(D_n, \sigma) \leq 3w$. Thus we can decide whether (S_n, τ) is explorable in time $\mathcal{O}(w^3 2^{3w} \Lambda)$ by applying the algorithm of Theorem 4.3.2 to (D_n, σ). ■

4.4 Win-win approach to regularly spaced times

In this section we will find necessary conditions for edge-explorability of temporal graphs with respect to their interval-membership-width. This will allow us to conclude that either we are given a no-instance or that the interval-membership-width is small (in which case we can employ our algorithmic results from the previous section).

We will apply this bidimensional approach to a variants of TEMPEULER(k) and STAREXP(k) in which we are given upper and lower bounds (u and ℓ respectively) on the difference between any two consecutive times at any edge. Specifically we will show that STAREXP(k) is in FPT parameterized by k, ℓ and u (Theorem 4.4.3) and that TEMPEULER(k) is in FPT parameterized by k and u (Theorem 4.4.4). In other words, these results allow us to trade in the dependences on the interval-membership-width of Corollary 4.3.4 and Theorem 4.3.2 for a dependences on k, ℓ, u and k, u respectively.

We note that, for STAREXP instances, the closer ℓ and u get, the more restricted the structure becomes to the point that the dependence on ℓ and u in the running time of our algorithm vanishes when $\ell = u$. In particular this shows that the problem of determining the explorability of STAREXP(k)-instances for which consecutive times at each edge are exactly λ time-steps apart (for some $\lambda \in \mathbb{N}$) is in FPT parameterized solely by k (Corollary 4.4.5). This partially resolves an open problem of Akrida, Mertzios and Spirakis [3] which asked to determine the complexity of exploring STAREXP(k)-instances with evenly-spaced times.

Towards these results, we will first provide sufficient conditions for non-explorability of any STAREXP(k) instance (Lemma 4.4.1). These conditions will depend only on: (1) knowledge of the maximum and minimum differences between any two successive appearances of any edge, (2) the interval-membership-width and (3) the maximum number of appearances k of any edge.

Lemma 4.4.1. *Let (S_n, τ) be a temporal star with at most k times at any edge and such that every two consecutive times at any edge differ at least by ℓ and at most by u. If (S_n, τ) is explorable, then $\mathbf{imw}(S_n, \tau) \leq (2(k-1)u+1)/(\ell+1)$.*

Proof. Let Λ be the lifetime of (S_n, τ), let $(F_t)_{t\in[\Lambda]}$ be the interval membership sequence of (S_n, τ) and choose

any $n \in [\Lambda]$ such that $|F_n| = \mathbf{imw}(S_n, \tau)$. Let m and M be respectively the earliest and latest times at which there are edges in F_n which are active and chose representatives e_m and e_M in F_n such that $m = \min \tau(e_m)$ and $M = \max \tau(e_M)$.

Recall that visiting any edge e in S_n requires us to us pick two appearances (which differ by at least $\ell + 1$ time-steps) of e (one appearance to go along e from the center of S_n to the leaf and another appearance to return to the center of the star). Thus, whenever we specify how to visit an edge e of F_n, we remove at least $\ell + 1$ time-steps from the available time-set $\{m, \dots, M\}$ at which any other edge in F_n can be visited (we need one time-step to travel to the leaf incident with e and then - after ℓ time-steps - we return to the center). Furthermore, since any exploration of (S_n, τ) must explore all of the edges in F_n, for (S_n, τ) to be explorable, we must have $|F_n|(\ell + 1) \leq M - m + 1$. This concludes the proof since $\mathbf{imw}(S_n, \tau) = |F_n|$ and $M - m \leq |\max \tau(e_M) - \min \tau(e_M)| + |\max \tau(e_m) - \min \tau(e_m)|$ (since, by the definition of F_n, n is in the intervals of any two elements of F_n) which is at most $2(k-1)u + 1$ (since consecutive times at any edge differ by at most u). ■

Notice that nearly-identical arguments yield the following slightly weaker result with respect to the TEMPEULER(k) problem.

Lemma 4.4.2. *Let (G, τ) be a* TEMPEULER(k) *instance such that every two consecutive times at any edge differ at most by u. If (G, τ) is temporally Eulerian, then* $\mathbf{imw}(G, \tau) \leq 2(k-1)u + 1$.

The reason that the we can only bound $\mathbf{imw}(G, \tau)$ above by $2(k-1)u+1$ (rather than $(2(k-1)u+1)/(\ell+1)$ as in the STAREXP(k) case of Lemma 4.4.1) is that temporal Euler circuits only visit each edge once (so exploring each edge only removes exactly one available time).

Lemma 4.4.1 allows us to employ a 'win-win' approach for STAREXP(k) when we know the maximum difference between consecutive times at any edge: either the considered instance does not satisfy the conditions of Lemma 4.4.1 (in which case we have a no-instance) or the interval-membership-width is small enough for us to usefully apply Corollary 4.3.4. These ideas allow us to conclude the following result. We point out that in the following Theorems 4.4.3, and 4.4.4 Corollary 4.4.5, we can drop the factor n in the running times if we assume that the relevant interval-membership-sequences are given.

Theorem 4.4.3. *Let (S_n, τ) be a temporal star with at most k times at any edge and such that every two consecutive times at any edge differ at least by ℓ and at most by u. There is an algorithm deciding whether*

(S_n, τ) is explorable in time $(2^{\mathcal{O}(ku/\ell)} + n)\Lambda$ where Λ is the lifetime of the input.

Proof. The algorithm proceeds as follows. First determine $\mathbf{imw}(S_n, \tau)$ (this can be done in time $\mathcal{O}(\Lambda n)$ where Λ is the lifetime of the input). If $\mathbf{imw}(S_n, \tau) > (2(k-1)u+1)/(\ell+1)$, then (S_n, τ) is not explorable by Lemma 4.4.1. Otherwise run the algorithm given in Corollary 4.3.4. In this case, since $w := \mathbf{imw}(S_n, \tau) \leq (2(k-1)u+1)/(\ell+1)$, we know that the algorithm of Corollary 4.3.4 will run on (S_n, τ) in time $2^{\mathcal{O}(ku/\ell)}\Lambda$. ■

Once again arguing by bidimensionality (this time using Lemma 4.4.2 and Theorem 4.3.2) we can deduce the following fixed-parameter tractability result for TEMPEULER.

Theorem 4.4.4. *Let (G, τ) be a* TEMPEULER(k) *instance such that every two consecutive times at any edge differ at most by u. There is an algorithm deciding whether (G, τ) is temporally Eulerian in time $(2^{\mathcal{O}(ku)} + n)\Lambda$ where Λ is the lifetime of the input.*

As a special case of Theorem 4.4.3 (i.e. the case where $\ell = u$) we resolve an open problem of Akrida, Mertzios and Spirakis [3] which asked to determine the complexity of exploring STAREXP(k)-instances with evenly-spaced times. In particular we show that the problem of deciding the explorability of such evenly-spaced STAREXP(k)-instances is in FPT when parameterized by k.

Corollary 4.4.5. *There is an algorithm which, given any* STAREXP(k) *instance (S_n, τ) with lifetime Λ and in which every two pairs of consecutive times assigned to any edge differ by the same amount, decides whether (S_n, τ) is explorable in time $(2^{\mathcal{O}(k)} + n)\Lambda$.*

4.5 Discussion

In this chapter we introduced a natural temporal analogue of Eulerian circuits and proved that, in contrast to the static case, TEMPEULER(k) is NP-complete for all $k \geq 2$ (where the $k = 2$ case – which we had left open [18] – was proven by Marino and Silva [75]). In fact we showed that the problem remains hard even when the underlying static graph has path-width 2, feedback vertex number 1 or vertex cover number 2 (Section 4.2). Along the way, we resolved an open problem of Akrida, Mertzios and Spirakis [3] by showing that STAREXP(k) is NP-complete for all $k \geq 4$. This result yields a complete complexity dichotomy with respect to k when combined with Akrida, Mertzios and Spirakis' results [3].

Our hardness results rule out FPT algorithms for TEMPEULER(k) and STAREXP(k) with respect to many standard parameters describing the structure of the underlying graph (such as path-width, feedback vertex

number and vertex-cover number). Motivated by these resutls, we introduced a new width measure which captures structural information that is purely temporal; we call this the *interval-membership-width*. In contrast to our hardness results, we showed that TEMPEULER(k) and STAREXP(k) can be solved in times $\mathcal{O}(w^3 2^w \Lambda)$ and $\mathcal{O}(w^3 2^{3w} \Lambda)$ respectively where w is our new parameter and Λ is the lifetime of the input.

Our fixed-parameter-tractability results parameterized by interval-membership-width can also be leveraged via a win-win approach to obtain tractability results for both TEMPEULER(k) and STAREXP(k) parameterized solely by k and the minimum and maximum differences between any two successive times in a time-set of any edge. These resutls allow us to partially resolve another open problem of Akrida, Mertzios and Spirakis concerning the complexity of STAREXP(k): we showed that it can be solved in time $2^{\mathcal{O}(k)}\Lambda$ when the input has evenly spaces appearances of each edge and lifetime Λ. We note, however, that it remains an open problem to determine the complexity of the evenly-spaced STAREXP(k) problem when k is unbounded.

Finally we point out that all of our hardness reductions hold also for the case of non-strict temporal walks and, with slightly more work, even our tractability results can be seen to hold for the non-strict case.

5 | Future work

This thesis proposed recursive decomposition methods and their associated width-measures in the settings of directed graphs, spined categories and temporal graphs.

In Chapter 2 we introduced *directed branch-width*, an analogue of tree-width for directed graphs. We found that the model-checking problem for a restricted variant of monadic second-order logic is in FPT parameterized by this measure. In particular, we showed that many NP-hard problems (including the directed versions of Hamilton path and Max-Cut) are tractable on digraph classes of bounded directed branch-width. This is particularly relevant since the directed variants of both the Hamilton Path and Max-Cut problems are NP-hard when parameterized by any of the other known 'tree-width-inspired' or 'rank-width-inspired' measures. Furthermore, on the structural side, we established a characterization of classes of bounded directed branch-width in terms of the rank-width of their line-graphs. This result is thus of independent interest since it provides a directed analogue of a theorem of Gurski and Wanke for undirected graphs [53].

There are many questions that arise from the introduction of directed branch-width. On the structural side, a natural question is to enquire whether we can push our method of obtaining new width-measures by taking iterated line-graphs any further; for example, can we characterize those classes $\mathcal{C}$ of digraphs:

- whose associated class $\vec{L}(\mathcal{C})$ of directed line-graphs has bounded *directed branch-width* or
- which are directed line-graphs of classes of bounded bi-cut-rank-width (i.e. $\mathcal{C} = \vec{L}(\mathcal{D})$ for some class $\mathcal{D}$ of bounded bi-cut-rank-width)?

This seems like a promising direction for further work since the line-graph operator is not too forgetful: for example it maps small vertex-cuts to cuts of low rank (as we observed in Chapter 2) and it is well-behaved with respect to spectral properties [50].

As we noted in Chapter 2, another direction for future work has to do with the study of the *tangles* (these represent regions of high connectivity in a graph) which are dual to directed branch-width. Giannopoulou, Kawarabayashi, Kreutzer and Kwon [49] also introduced a notion under the name of 'directed tangle' which, as we already noted, is completely distinct from the tangles arising from directed branch-width. It is thus an interesting avenue for further research to compare our notion to theirs.

On the algorithmic side, our work on directed branch-width alludes to an interesting narrative around the algorithmic power of width-measures on directed graphs which we will sketch now. Recall that Ganian et. al. showed [47] (see Theorem 2.2.10) that any directed width-measure which is:

- closed under directed topological minors and
- which is not tree-width bounding

cannot be algorithmically powerful (in the sense that, subject to standard complexity-assumptions, the $\mathbf{MSO}_1$-model-checking problem cannot admit an XP algorithm parameterized by such a measure). This result seems to paint a hopeless picture for the quest of obtaining algorithmically useful width-measures for digraphs. Yet we were able to obtain strong – albeit not as strong as what is precluded by Theorem 2.2.10 – algorithmic results parameterized by directed branch-width. Obviously there is no danger of contradictions here since directed branch-width is not closed under directed topological minors (Theorem 2.4.15). However, this contrast alludes (in an intuitive and speculative way) to another answer for why we are able to obtain our results in spite of Ganian et. al.'s negative algorithmic meta-theorem: directed branch-width is based upon what might be considered the least-directed of all the directed connectivity functions. In fact, although directed branch-width is not tree-width bounding, it differs from underlying tree-width only at sources and sinks (Corollary 2.4.8). In contrast, although all of the tree-width-inspired measures make use of weaker connectivity functions (think of weak vs strong connectivity in digraphs) they suffer algorithmic shortcoming that directed branch-width does not (in fact they are not algorithmically powerful in the sense of Ganian et. al.'s results [47]). Thus it seems that these considerations allude to a trade-off between the 'degree of directedness' of the connectivity-notion one chooses to work with and the algorithmic power of the related width-measure. In this context it would thus be an interesting, but quite open-ended direction for future work to find ways of formalising the notion of 'directedness' of a connectivity function allowing for a thorough study of the relationships between the choice of a notion of connectivity and the algorithmic power that the associated width-measures allow us.

In Chapter 3 we introduced the notions of *spined categories* and *spined functors*. This allowed us

to provide formal, categorical definitions (see diagram below) of the terms ‘class of decomposable objects’, ‘recursive decomposition’ and its associated ‘width measure’ which had so far been used more as informal umbrella terms.

‘class of recursively decomposable objects’	— formalized as →	spined category
‘recursive decomposition’	— formalized as →	(pseudo)-chordal completion
‘width measure’	— formalized as →	triangulation functor

To do this, we introduced S-functors, which are a vast generalization of Halin’s S-functions. In fact, analogously to Halin’s result (Theorem 3.1.2), we showed that S-functors form an upper-complete semi-lattice with the triangulation functor (our abstract analogue of tree-width) as the maximum element in any measurable spined category (Theorems 3.4.13 and 3.4.15). The study of S-functors over arbitrary spined categories is an interesting avenue for further enquiry which we believe requires more attention (even just for the case of graphs).

Our categorical characterization of tree decompositions is framed in terms of pseudo-chordal and chordal completions. Surprisingly, we found that, from the point of view of defining an abstract analogue of tree-width, the choice between pseudo-chordal and chordal completions is inconsequential since their two associated S-functors (Δ and Δ^{ch} respectively) always coincide. However, we point out that, despite having a complete understanding of chordal objects in $\mathbf{Gr}_{mono}$ (recall that these turned out to be chordal graphs; see Example 3.4.6), even in this familiar category, we do not yet have a characterization of pseudo-chordal objects. This is an interesting open problem whose solution should also deepen our understanding of S-functors (since all S-functors agree on pseudo-chordal objects).

We note that our algebraic formulation of tree-width allows for great generality since it can be thought as a black-box associating to each measurable spined category an appropriate analogue of tree-width. Thus we expect that it will be useful for defining new tree-width analogues in the future. Of particular interest is the broad question of finding more algebraic instantiations of spined categories. Can we find good notions of arrow (i.e. satisfying Property **SC1**) allowing us to find examples of measurable spined categories over groups or posets or other algebraic objects? And, if so, what algorithmic applications will their respective triangulation functors allow?

In the context of this thesis, it is obvious to ask whether we can use the theory of spined categories to define an algorithmically-useful temporal analogue of tree-width. We will address this question briefly. Note

that in general spined categories have a clarifying effect on the question of finding an algorithmically useful tree-width analogue since they reduce this vague question to the following two concrete tasks:

(Q1) finding a containment relation satisfying Property **SC1** and

(Q2) finding a suitable proxy-pushout operation satisfying Property **SC2**.

There are many possible ways of defining containment relations on temporal graphs. However, for any such notion to be algorithmically useful, we need it to take into account not only the underlying static structure, but also *temporal* information. This is indeed a fruitful line for future work, but it is beyond the scope of this thesis since even the comparatively simple problem of finding a good[1] notion of temporal subgraph is a challenging question that will likely require new theoretical tools and a deeper mathematical and philosophical understanding of temporal graphs.

In Chapter 4 we introduced a purely temporal width measure called *interval-membership-width*. This width measure is related to the path-width of an associated static graph and it is defined via a related decomposition called the *interval-membership-sequence*. Interval-membership-width turns out to be algorithmically useful even for problems that remain hard on instances whose underlying static graph has tree-width at most 2 or vertex-cover number at most 2. In fact we provide linear-time algorithms parameterized by our new measure for the problems of: (1) determining whether a temporal graph is temporally Eulerian and (2) determining the explorability of temporal stars. We found that both of these problems are NP-complete as soon as we allow edges to have respectively 2 (where Marino and Silva solved the $k = 2$ case that we left open) and 4 temporal appearances. Furthermore, recall that these results yield complexity dichotomies since, for fewer allowed-appearances, both problems are polynomial-time solvable.

We believe that interval-membership-width and interval-membership-sequences will be useful in the formulation of dynamic-programming algorithms for other problems on temporal graphs. Likely candidates are decision (or counting) problems related to temporal exploration and/or connectivity. Furthermore, there are many techniques available in the parameterized complexity literature (e.g. see Cygan et al. [29]) designed for the improvement of running times of dynamic programming algorithms on tree decompositions. Given the similarities, it is reasonable to expect for it to be possible to port many of these ideas to the setting of designing even faster dynamic programming routines running on temporal graphs and their interval-membership sequences.

[1] Spined categories can help us decide whether we have found a good notion based on how algorithmically powerful the associated notion of tree-width is.

A more open-ended question that is raised from our work on temporal graphs has to do with finding ways of defining other kinds of temporal structure. Interval-membership-width can be thought of as describing temporal structure with low temporal connectivity: as we mentioned, we can associate each temporal graph (G, τ) to the interval graph

$$\mathcal{I}(G, \tau) = (E(G), \{ef : \{\min \tau(e), \dots, \max \tau(e)\} \cap \{\min \tau(f), \dots, \max \tau(f)\} \neq \emptyset\})$$

such that the tree-width (or equivalently, since it is an interval graph, the path-width) of $\mathcal{I}(G, \tau)$ is $\mathbf{tw}(\mathcal{I}(G, \tau)) = \mathbf{imw}(G, \tau) - 1$. In this light, it would be interesting to study other mappings (or functors) from (an appropriately defined category of) temporal graphs to other structures (static graphs, or partial orders ...) which we understand better. Depending on the structure that is preserved by such mappings, we might be able to more eloquently define notions of temporal structure which are currently out of our immediate reach. We note for example that Fluschnik et. al. [44] started exploring ideas of this kind by proposing a temporal analogue of tree-width which is defined in terms of the underlying tree-width of an associated directed graph (which they call a *static expansion*). To the best of our knowledge, this width measure has not been applied further and, in fact, our general idea of studying the structure of temporal graphs via auxiliary, better understood objects seems to be understudied.

Bibliography

[1] I. Adler, G. Gottlob, and M. Grohe. Hypertree width and related hypergraph invariants. *European Journal of Combinatorics*, 28(8):2167 – 2181, 2007. URL: https://doi.org/10.1016/j.ejc.2007.04.013.

[2] E. C. Akrida, G. B. Mertzios, S. Nikoletseas, C. Raptopoulos, P. G. Spirakis, and V. Zamaraev. How fast can we reach a target vertex in stochastic temporal graphs? *Journal of Computer and System Sciences*, 114:65–83, 2020. URL: https://doi.org/10.1016/j.jcss.2020.05.005.

[3] E. C. Akrida, G. B. Mertzios, and P. G. Spirakis. The temporal explorer who returns to the base. In P. Heggernes, editor, *Algorithms and Complexity, CIAC 2019*, pages 13–24, Cham, 2019. Springer International Publishing. URL: https://doi.org/10.1007/978-3-030-17402-6_2.

[4] O. Amini, F. Mazoit, N. Nisse, and S. Thomassé. Submodular partition functions. *Discrete Mathematics*, 309(20):6000–6008, 2009. URL: https://doi.org/10.1016/j.disc.2009.04.033.

[5] S. Awodey. *Category theory*. Oxford university press, 2010.

[6] K. Axiotis and D. Fotakis. On the Size and the Approximability of Minimum Temporally Connected Subgraphs. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 149:1–149:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: https://doi.org/10.4230/LIPIcs.ICALP.2016.149.

[7] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: Membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 44–54, New York, NY, USA, 2006. Association for Computing Machinery. URL: https://doi.org/10.1145/1150402.1150412.

[8] J. Bang-Jensen and G. Gutin. *Classes of directed graphs*. Springer, 2018.

[9] J. Bang-Jensen and G. Z. Gutin. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008.

[10] K. A. Berman. Vulnerability of scheduled networks and a generalization of menger's theorem. *Networks: An International Journal*, 28(3):125–134, 1996.

[11] U. Bertelè and F. Brioschi. *Nonserial dynamic programming*. Academic Press, Inc., 1972. URL: https://doi.org/10.1016/s0076-5392(08)x6010-2.

[12] P. Berthomé, T. Bouvier, F. Mazoit, N. Nisse, and R. Pardo Soares. A unified FPT algorithm for width of partition functions. Research Report RR-8372, INRIA, September 2013. URL: https://hal.inria.fr/hal-00865575.

[13] D. Berwanger, A. Dawar, P. Hunter, S. Kreutzer, and J. Obdržálek. The DAG-width of directed graphs. *Journal of Combinatorial Theory, Series B*, 102(4):900–923, 2012. URL: https://doi.org/10.1016/j.jctb.2012.04.004.

[14] S. Bhadra and A. Ferreira. Complexity of connected components in evolving graphs and the computation of multicast trees in dynamic networks. In *ADHOC-NOW 2003*, pages 259–270, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-540-39611-6_23.

[15] H. L. Bodlaender and D. M. Thilikos. Constructive linear time algorithms for branchwidth. In *International Colloquium on Automata, Languages, and Programming*, pages 627–637. Springer, 1997. URL: https://doi.org/10.1007/3-540-63165-8_217.

[16] André Bouchet. Connectivity of isotropic systems. *Annals of the New York Academy of Sciences*, 555(1):81–93, 1989. URL: https://doi.org/10.1111/j.1749-6632.1989.tb22439.x.

[17] B. M. Bumpus and Z. A. Kocsis. Spined categories: generalizing tree-width beyond graphs, 2021.

[18] B. M. Bumpus and K. Meeks. Edge exploration of temporal graphs. *arXiv preprint arXiv:2103.05387*, 2021.

[19] B. M. Bumpus, K. Meeks, and W. Pettersson. Directed branch-width: A directed analogue of tree-width. *arXiv preprint arXiv:2009.08903*, 2020.

[20] J. Carmesin, R. Diestel, M. Hamann, and F. Hundertmark. *k*-blocks: A connectivity invariant for graphs. *SIAM Journal on Discrete Mathematics*, 28(4):1876–1891, 2014. URL: 10.1137/130923646.

[21] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. In H. Frey, X. Li, and S. Ruehrup, editors, *Ad-hoc, Mobile, and Wireless Networks*, pages 346–359, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. URL: https://doi.org/10.1080/17445760.2012.668546.

[22] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012. URL: https://doi.org/10.1080/17445760.2012.668546, arXiv:https://doi.org/10.1080/17445760.2012.668546.

[23] A. Casteigts, A.-S. Himmel, H. Molter, and P. Zschoche. Finding Temporal Paths Under Waiting Time Constraints. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation (ISAAC 2020)*, volume 181 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:18, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: https://doi.org/10.4230/LIPIcs.ISAAC.2020.30.

[24] D. G. Corneil and U. Rotics. On the relationship between clique-width and treewidth. In Andreas Brandstädt and Van Bang Le, editors, *Graph-Theoretic Concepts in Computer Science*, pages 78–90, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. URL: https://doi.org/10.1007/3-540-45477-2_9.

[25] B. Courcelle. The monadic second-order logic of graphs i. recognizable sets of finite graphs. *Information and computation*, 85(1):12–75, 1990. URL: https://doi.org/10.1016/0890-5401(90)90043-H.

[26] B. Courcelle and J. Engelfriet. *Graph structure and monadic second-order logic: a language-theoretic approach*, volume 138. Cambridge University Press, 2012. URL: https://doi.org/10.1017/CBO9780511977619.

[27] B. Courcelle, J. Engelfriet, and G. Rozenberg. Handle-rewriting hypergraph grammars. *Journal of computer and system sciences*, 46(2):218–270, 1993. URL: https://doi.org/10.1016/0022-0000(93)90004-G.

[28] B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000. URL: https://doi.org/10.1007/s002249910009.

[29] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized algorithms*. Springer, 2015. URL: https://doi.org/10.1007/978-3-319-21275-3.

[30] R. Diestel. *Graph theory*. Springer, 2010.

[31] R. Diestel, F. Hundertmark, and S. Lemanczyk. Profiles of separations: in graphs, matroids, and beyond. *Combinatorica*, 39(1):37–75, 2019. URL: https://doi.org/10.1007/s00493-017-3595-y.

[32] R. Diestel and S.-i. Oum. Unifying duality theorems for width parameters in graphs and matroids. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 1–14. Springer, 2014. URL: https://doi.org/10.1007/978-3-319-12340-0_1.

[33] R. Diestel and S.-i. Oum. Tangle-tree duality in abstract separation systems. *arXiv preprint arXiv:1701.02509*, 2017.

[34] G. A. Dirac. On rigid circuit graphs. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, volume 25, pages 71–76. Springer, 1961. URL: https://doi.org/10.1007/BF02992776.

[35] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness ii: On completeness for w[1]. *Theoretical Computer Science*, 141(1):109 – 131, 1995. URL: https://doi.org/10.1016/0304-3975(94)00097-3.

[36] J. Enright, K. Meeks, G. B. Mertzios, and V. Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. In P. Rossmanith, P. Heggernes, and J.-P. Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, volume 138 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 57:1–57:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: 10.4230/LIPIcs.MFCS.2019.57.

[37] J. Enright, K. Meeks, G. B. Mertzios, and V. Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. *Journal of Computer and System Sciences*, 119:60–77, 2021. URL: https://doi.org/10.1016/j.jcss.2021.01.007.

[38] J. Enright, K. Meeks, and F. Skerman. Assigning times to minimise reachability in temporal graphs. *Journal of Computer and System Sciences*, 115:169–186, 2021. URL: https://doi.org/10.1016/j.jcss.2020.08.001.

[39] Jessica Enright and Rowland Raymond Kao. Epidemics on dynamic networks. *Epidemics*, 24:88–97, 2018. URL: https://doi.org/10.1016/j.epidem.2018.04.003.

[40] T. Erlebach, M. Hoffmann, and F. Kammer. On temporal graph exploration. In M. M. Halldórsson, K. Iwama, N. Kobayashi, and B. Speckmann, editors, *ICALP 2015*, pages 444–455, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-662-47672-7_36.

[41] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, pages 128–140, 1741.

[42] M. R. Fellows and M. A. Langston. Nonconstructive tools for proving polynomial-time decidability. *J. ACM*, 35(3):727–739, 1988. URL: https://doi.org/10.1145/44483.44491.

[43] J. Flum and M. Grohe. Parameterized complexity theory. 2006. *Texts Theoret. Comput. Sci. EATCS Ser*, 2006. URL: https://doi.org/10.1007/3-540-29953-X.

[44] T. Fluschnik, H. Molter, R. Niedermeier, M. Renken, and P. Zschoche. As time goes by: Reflections on treewidth for temporal graphs. In *Treewidth, Kernels, and Algorithms: Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, pages 49–77. Springer International Publishing, Cham, 2020. URL: https://doi.org/10.1007/978-3-030-42071-0_6.

[45] F. V. Fomin, P. A Golovach, D. Lokshtanov, and S. Saurabh. Intractability of clique-width parameterizations. *SIAM Journal on Computing*, 39(5):1941–1956, 2010. URL: 10.1137/080742270.

[46] B. Fong and D. I Spivak. *An invitation to applied category theory: seven sketches in compositionality*. Cambridge University Press, 2019.

[47] R. Ganian, P. Hliněný, J. Kneis, D. Meister, J. Obdržálek, P. Rossmanith, and S. Sikdar. Are there any good digraph width measures? In *International Symposium on Parameterized and Exact Computation*, pages 135–146. Springer, 2010. URL: https://doi.org/10.1007/978-3-642-17493-3_14.

[48] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman, San Francisco, 1979.

[49] A. C. Giannopoulou, K i. Kawarabayashi, S. Kreutzer, and O j. Kwon. The canonical directed tree decomposition and its applications to the directed disjoint paths problem, 2020. arXiv:2009.13184.

[50] C. Godsil and G. F. Royle. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2001. URL: https://doi.org/10.1007/978-1-4613-0163-9.

[51] M. C. Golumbic. Chapter 4 - triangulated graphs. In *Algorithmic Graph Theory and Perfect Graphs*, pages 81–104. Academic Press, 1980. URL: https://doi.org/10.1016/B978-0-12-289260-8.50011-X.

[52] M. Grohe. Descriptive complexity, canonisation, and definable graph structure theory, cambridge university press, cambridge, 2017, x + 544 pp. *The Bulletin of Symbolic Logic*, 23(4):493–494, 2017.

[53] F. Gurski and E. Wanke. Line graphs of bounded clique-width. *Discrete Mathematics*, 307(22):2734–2754, 2007. URL: https://doi.org/10.1016/j.disc.2007.01.020.

[54] F. Gurski, E. Wanke, and E. Yilmaz. Directed NLC-width. *Theoretical Computer Science*, 616:1–17, 2016. URL: https://doi.org/10.1016/j.tcs.2015.11.003.

[55] R. Halin. S-functions for graphs. *Journal of Geometry*, 8(1-2):171–186, 1976. URL: https://doi.org/10.1007/BF01917434.

[56] F. Harary and G. Gupta. Dynamic graph models. *Mathematical and Computer Modelling*, 25(7):79–87, 1997. URL: https://doi.org/10.1016/S0895-7177(97)00050-2.

[57] P. Holme and J. Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, 2012. Temporal Networks. URL: https://doi.org/10.1016/j.physrep.2012.03.001.

[58] Petter Holme. Modern temporal network theory: a colloquium. *The European Physical Journal B*, 88(9):1–30, 2015. URL: https://doi.org/10.1140/epjb/e2015-60657-4.

[59] P. Hunter and S. Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. *Theoretical Computer Science (TCS)*, 399, 2008. URL: https://doi.org/10.1016/j.tcs.2008.02.038.

[60] OEIS Foundation Inc. The on-line encyclopedia of integer sequences. http://oeis.org/A051903, 2020. Accessed: 2020-11-08.

[61] Vít Jelínek. The rank-width of the square grid. *Discrete Applied Mathematics*, 158(7):841–850, 2010. URL: https://doi.org/10.1016/j.dam.2009.02.007.

[62] D. S. Johnson. The np-completeness column: an ongoing guide. *Journal of Algorithms*, 6(3):434 – 451, 1985. URL: https://doi.org/10.1016/0196-6774(85)90012-4.

[63] T. Johnson, N. Robertson, P. D. Seymour, and R. Thomas. Directed tree-width. *Journal of combinatorial theory. Series B*, 82(1):138–154, 2001. URL: https://doi.org/10.1006/jctb.2000.2031.

[64] M. M. Kanté and M. Rao. $\mathbb{F}$-rank-width of (edge-colored) graphs. In *International Conference on Algebraic Informatics*, pages 158–173. Springer, 2011. URL: https://doi.org/10.1007/978-3-642-21493-6_10.

[65] D. Kempe and J. Kleinberg. Protocols and impossibility results for gossip-based communication mechanisms. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 471–480, 2002. URL: https://doi.org/10.1109/SFCS.2002.1181971.

[66] D. Kempe, J. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002. URL: https://doi.org/10.1006/jcss.2002.1829.

[67] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, page 137–146, New York, NY, USA, 2003. Association for Computing Machinery. URL: https://doi.org/10.1145/956750.956769.

[68] S. Kreutzer and O.-j. Kwon. Digraphs of bounded width. In *Classes of Directed Graphs*, pages 405–466. Springer, 2018. URL: https://doi.org/10.1007/978-3-319-71840-8_9.

[69] M. Lampis, G. Kaouri, and V. Mitsou. On the algorithmic effectiveness of digraph decompositions and complexity measures. In *International Symposium on Algorithms and Computation*, pages 220–231. Springer, 2008. URL: https://doi.org/10.1016/j.disopt.2010.03.010.

[70] T. Leinster. Codensity and the ultrafilter monad. *Theory and Applications of Categories*, (28):332–370, 3 2013.

[71] Tom Leinster. The categorical origins of Lebesgue integration. *arXiv e-prints*, page arXiv:2011.00412, October 2020. arXiv:2011.00412.

[72] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 1(1):2–es, March 2007. URL: https://doi.org/10.1145/1217299.1217301.

[73] L. Libkin. *Elements of finite model theory*. Springer Science & Business Media, 2013. URL: https://doi.org/10.1007/978-3-662-07003-1.

[74] L. Lovász. Graph minor theory. *Bull. Amer. Math. Soc. 43 (2006), 75-86*, 43(1), 2005. URL: https://doi.org/10.1090/S0273-0979-05-01088-8.

[75] A. Marino and A. Silva. K\"{o} nigsberg sightseeing: Eulerian walks in temporal graphs. *arXiv preprint arXiv:2103.07522*, 2021.

[76] G. B Mertzios, O. Michail, and P. G Spirakis. Temporal network optimization subject to connectivity constraints. *Algorithmica*, 81(4):1416–1449, 2019. URL: https://doi.org/10.1007/s00453-018-0478-6.

[77] G. B Mertzios, H. Molter, R. Niedermeier, V. Zamaraev, and P. Zschoche. Computing maximum matchings in temporal graphs. *arXiv preprint arXiv:1905.05304*, 2019.

[78] G. B Mertzios, H. Molter, R. Niedermeier, V. Zamaraev, and P. Zschoche. Computing maximum matchings in temporal graphs. *arXiv preprint arXiv:1905.05304*, 2019.

[79] O. Michail. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, 12(4):239–280, 2016. URL: https://doi.org/10.1080/15427951.2016.1177801.

[80] O. Michail and P. G. Spirakis. Traveling salesman problems in temporal graphs. In E. Csuhaj-Varjú, M. Dietzfelbinger, and Z. Ésik, editors, *MFCS 2014*, pages 553–564, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. URL: https://doi.org/10.1016/j.tcs.2016.04.006.

[81] S.-i. Oum. Rank-width is less than or equal to branch-width. *Journal of Graph Theory*, 57(3):239–244, 2008. URL: https://doi.org/10.1002/jgt.20280.

[82] S.-i. Oum. Rank-width: Algorithmic and structural results. *Discrete Applied Mathematics*, 231:15–24, 2017. URL: https://doi.org/10.1016/j.dam.2016.08.006.

[83] S.-i. Oum and P. D. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006. URL: https://doi.org/10.1016/j.jctb.2005.10.006.

[84] E. Riehl. *Category theory in context*. Courier Dover Publications, 2017.

[85] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 9 1986. URL: 10.1016/0196-6774(86)90023-4.

[86] N. Robertson and P. D. Seymour. Graph minors V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(1):92–114, 8 1986. URL: 10.1016/0095-8956(86)90030-4.

[87] N. Robertson and P. D. Seymour. Graph minors X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991. URL: https://doi.org/10.1016/0095-8956(91)90061-n.

[88] N. Robertson and P.D. Seymour. Graph minors. iii. planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984. URL: https://doi.org/10.1016/0095-8956(84)90013-3.

[89] N. Robertson and P.D. Seymour. Graph minors. xx. wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325 – 357, 2004. URL: https://doi.org/10.1016/j.jctb.2004.08.001.

[90] A.-S. Ruget, G. Rossi, P. T. Pepler, G. Beaunée, C. J. Banks, J. Enright, and R. R. Kao. Multi-species temporal network of livestock movements for disease spread. *Applied Network Science*, 6(1):1–20, 2021. URL: https://doi.org/10.1007/s41109-021-00354-x.

[91] M. A. Safari. D-width: A more natural measure for directed tree width. In *International Symposium on Mathematical Foundations of Computer Science*, pages 745–756. Springer, 2005. URL: https://doi.org/10.1007/11549345_64.

[92] P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994. URL: https://doi.org/10.1007/bf01215352.

[93] M. Vatshelle. New width parameters of graphs. 2012.

[94] H. Whitney. Congruent graphs and the connectivity of graphs. In *Hassler Whitney Collected Papers*, pages 61–79. Springer, 1992. URL: https://doi.org/10.1007/978-1-4612-2972-8_4.

[95] H. Wu, J. Cheng, Y. Ke, S. Huang, Y. Huang, and H. Wu. Efficient algorithms for temporal path computation. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2927–2942, 2016. URL: 10.1109/TKDE.2016.2594065.

[96] B Bui Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285, 2003. URL: https://doi.org/10.1142/S0129054103001728.

Index