FPT Parameterisations of Fractional and Generalised Hypertree Width

MATTHIAS LANZINGER*, TU Wien, Austria
IGOR RAZGON, Durham University, United Kingdom
DANIEL UNTERBERGER*, TU Wien, Austria

We present the first fixed-parameter tractable (FPT) algorithms for exact computation of generalized hypertree width (ghw) and fractional hypertree width (fhw). Our algorithms are parameterized by the target width, the rank, and the maximum degree of the input hypergraph. More generally, we show that testing f-width is in FPT for a broad class of width functions that we call *manageable*. This class contains the edge cover number ρ and its fractional relaxation ρ^*, and thus covers both generalized and fractional hypertree width. We additionally extend our framework to also obtain an fpt algorithm for computing a discretized version of adaptive width. Our approach extends a recent algorithm for treewidth (Bojańcyk & Pilipczuk, LMCS 2022) that utilises monadic second-order transductions.

To extend this idea beyond treewidth we develop new combinatorial machinery around elimination forests in hypergraphs, culminating in a structural normal form for optimal witnesses that makes transduction-based optimisation applicable in the much more general context of manageable width functions.

This yields the first exact FPT algorithms for these measures under any nontrivial parameterisation and provides structural tools that may enable more direct optimisation algorithms.

CCS Concepts: • **Theory of computation → Fixed parameter tractability**; **Database query processing and optimization (theory)**.

Additional Key Words and Phrases: hypertree decompositions, generalised hypertree width, fractional hypertree width, fixed parameter tractable

ACM Reference Format:

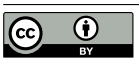
Matthias Lanzinger, Igor Razgon, and Daniel Unterberger. 2026. FPT Parameterisations of Fractional and Generalised Hypertree Width. *Proc. ACM Manag. Data* 4, 2 (PODS), Article 104 (May 2026), 17 pages. https://doi.org/10.1145/3801900

1 Introduction

A tree decomposition of a hypergraph H is a tree T with a function bag which assigns each vertex in T a set of vertices of H such that certain conditions hold. The *f-width* of a tree decomposition (T, bag) is defined as $\max_{u \in V(T)} f(\text{bag}(u))$. This generalises various widely studied (hyper)graph parameters in the literature. For instance, when f is the function $B \mapsto |B| - 1$, the f-width is treewidth [24], one of the most prominent concepts in the study of graph algorithms. When f is the edge cover number ρ of the bag, or its fractional relaxation ρ^*, the f-width equals generalized

*Supported by the Vienna Science and Technology Fund (WWTF) [10.47379/ICT2201].

Authors' Contact Information: Matthias Lanzinger, TU Wien, Vienna, Austria, matthias.lanzinger@tuwien.ac.at; Igor Razgon, Durham University, Durham, United Kingdom, igor.razgon@durham.ac.uk; Daniel Unterberger, TU Wien, Vienna, Austria, daniel.unterberger@tuwien.ac.at.



ACM 2836-6573/2026/5-ART104
https://doi.org/10.1145/3801900

hypertree width (ghw) [15] and fractional hypertree width (fhw) [16], respectively. These parameters have been shown central to the complexity of a wide range of problems whose instances are naturally represented as hypergraphs, e.g., conjunctive query evaluation [1, 14, 23], graph and hypergraph counting problems [6, 7, 13], problems in game theory [10], or combinatorial auctions [9].

Despite structural similarities to treewidth, deciding the ghw and fhw of a hypergraph is significantly harder. While deciding whether $\mathsf{tw}(H) \leq k$ for hypergraph H[1] is famously fixed-parameter tractable (FPT) when parameterised by the width k [2, 3], whereas the analogous problems for ghw and fhw are both paraNP-hard [13, 15]. Furthermore Gottlob et al. [11] gave an FPT reduction from the set cover problem to checking ghw which also implies that, in general, the problem is hard to approximate assuming $\mathrm{W}[1] \neq \mathrm{FPT}$ [17].

Recently significant progress has been made on identifying computationally easier fragments of these problems. Gottlob et al. [12, 13] showed that deciding whether $\mathsf{ghw}(H) \leq k$ and $\mathsf{fhw}(H) \leq k$ is in the complexity class XP when parameterised by the width k, i.e., permitting an $O(n^{f(k)})$ algorithm, under loose structural restrictions on H. Even more recently, Lanzinger and Razgon [19] gave the first FPT approximation algorithm for ghw for hypergraphs with bounded edge-intersection cardinality and in parallel Korchemna et al. [18] showed that even polynomial time approximation of ghw and fhw is possible under similar structural restrictions. Nonetheless, as of now, no fixed-parameter tractable algorithms under any non-trivial parametrisations are known for either problem.

In this paper we present the first exact FPT algorithms for deciding $\mathsf{ghw}(H) \leq k$ and $\mathsf{fhw}(H) \leq k$. Since the natural parametrization by width k alone is known to be paraNP-hard, obtaining FPT algorithms requires parameterising beyond width. Concretely, we additionally consider the rank (the maximum edge cardinality) and the maximum vertex degree (denoted by $\Delta(\cdot)$) of the input hypergraph as parameters. Formally we study the following class of problems.

f-WIDTH-CHECK

Input	Hypergraph H and $k > 0$.
Parameters	$\mathit{rank}(H)$, $\Delta(H)$ and k
Output	$f\text{-}\mathsf{width}(H) \leq k$?

In order to solve this question for ghw and fhw, we show that deciding f-width in general is FPT in this setting for a large class of functions f that we will call *manageable*. Informally, a manageable f is additive on non-adjacent disjoint sets and $f(U) \leq k$ implies bounds on the size of U, in addition to some minor technical conditions (the notion is introduced formally in Section 3). In particular, both *ρ and ρ^* are manageable width functions* and thus ghw and fhw are instances of f-width with manageable f. Our main result then is the following.

THEOREM 1. *f-WIDTH-CHECK is fixed parameter tractable for any manageable width function f.*

We approach this problem by building on the transduction-based treewidth algorithm of Bojańczyk and Pilipczuk [5]. At a high level, their result shows how to express the process of turning a potentially suboptimal tree decomposition into an optimal one by an MSO transduction, based on deep combinatorial insights on the fine structure of so-called elimination forests (see Section 2). In our setting, we follow the same blueprint: we construct an MSO transduction that, given an

[1]The treewidth of a hypergraph is equivalent to the treewidth of its primal graph, hence classical results on graphs transfer directly. However that even hypergraph with ghw 1 can have arbitrarily high treewidth and this is therefore not helpful in deciding many hypergraph width measures.

appropriate bounded-treewidth backbone, produces a candidate decomposition and an MSO test that the produced decomposition has f-width at most k.

The main difficulty in lifting this blueprint from treewidth to ghw/fhw is that the width function stops being "compositional". For treewidth, $f(U) = |U| - 1$ satisfies a strong separability property: for every partition $U = U_1 \uplus U_2$, the value $f(U)$ is determined by $f(U_1)$ and $f(U_2)$ (via a fixed monotone combination). The construction in [5] crucially exploits this kind of compositionality. In contrast, for $f \in \{\rho, \rho^*\}$, the value on U cannot in general be recovered from the values on an arbitrary partition. Our key insight is that a weaker form *does* hold for these functions: composition becomes possible once we restrict attention to the splits that are *edge-disjoint* (no hyperedge meets both sides), and a substantial part of the paper is devoted to developing the combinatorial machinery needed to make this restriction compatible with the transduction framework.

Our main technical contribution is the extension of the approach of [5] to width functions that only permit such weaker compositionality for edge-disjoint partitions. This requires large-scale changes from the treewidth result, combining multiple additional combinatorial insights with explicit abstraction of various implicit structural observations. A secondary motivation for isolating this structure is that it is not tied to MSO. Our combinatorial analysis concludes in a hypergraph dealternation principle (Reduced Hypergraph Dealternation) that shows that optimal witnesses are in a certain technical sense structurally well-behaved. As argued for treewidth in [5], such principles are exactly what underlies Bodlaender-Kloks-style direct FPT algorithms for treewidth. We therefore view our results as providing not only an MSO/transduction-based FPT route, but also a concrete combinatorial foundation for more direct and practical FPT algorithms for ghw/fhw.

As a final result, our framework produces an MSO test for f-width $\leq k$. This lets us go beyond a single f and handle certain *families* of width functions in one shot. As an illustration, we consider a discretized variant of *adaptive width* [21], defined as the $\mathcal{F}$-width where $\mathcal{F}_\delta$ ranges over independent-set functions whose vertex weights are multiples of $1/\delta$ (and every hyperedge has total weight at most 1). We show that this discretized adaptive width can be computed exactly in FPT time.

Theorem 2. *For each integer $\delta \geq rank(H)$, $\mathcal{F}_\delta$-width is fixed-parameter tractable parameterised by k, $rank(H)$, $\Delta(H)$, and δ.*

Theorem 2 leaves an interesting open question as to whether the discretized adaptive width can approximate the actual adaptive width. We provide some initial insight related to this question. We consider a relaxed version $\mathcal{F}^*_\delta$ of $\mathcal{F}_\delta$ consisting of all independent set functions where all the non-zero weights are at least δ (or, to put it differently, there are no small non-zero weights $0 < x < \delta$). We show that Theorem 2, combined with a simple rounding strategy, implies factor 2 FPT approximation of $\mathcal{F}^*_\delta$-width.

The rest of this paper is structured as follows. We formally define tree decompositions and key width measures, as well as further necessary technical preliminaries in Section 2. We present an extended technical overview of our main results and the central statements leading to it in Section 3. The lifting of our framework to sets of functions and the corresponding results for adaptive width are presented in Section 4. Closing remarks and discussion of a number of interesting new combinatorial open problems raised by our results are given in Section 5.

2 Preliminaries

We say a family $(U_i)_{i \in I}$ of sets is a *weak partition* of a set V if $\cup_{i \in I} U_i = V$ and $U_i \cap U_j = \emptyset$ for $i \neq j$. If no U_i is empty, we say it is a *partition* of V.

Graphs & Hypergraphs A *hypergraph* H is a tuple (V, E) where V is a finite set and E is a set of non-empty subsets $e \subseteq V$. If all $e \in E$ have exactly two elements, we refer to H as a (simple) graph.

We call V the vertices of G and E the edges of H. For a hypergraph H, we denote the vertices of H by $V(H)$ and the edges by $E(H)$. We write $I(v) := \{e \in E(H) \mid v \in e\}$ for the set of edges incident to vertex v, and we say two vertices $u \neq v$ are connected if $I(u) \cap I(v) \neq \emptyset$. For a vertex subset $U \subseteq V(H)$, we call the hypergraph H' with $V(H') = U$ and $E(H') = \{U \cap e : e \in E(H),\ U \cap e \neq \emptyset\}$ the *subgraph* of H *induced* by U, denoted by $H[U]$. For graphs, we only keep edges of cardinality 2 in the *induced subgraph*, i.e. only edges contained in U. We assume throughout that hypergraphs have no isolated vertices, i.e., every vertex in a hypergraph is contained in some edge.

Let G be a graph. A path from v_1 to v_ℓ in G is a sequence $(v_1, v_2, \cdots, v_\ell)$ of distinct vertices such that $\{v_i, v_{i+1}\} \in E$ for $1 \leq i \leq \ell - 1$. A cycle is a path except that $v_1 = v_\ell$. A graph is *connected* if, for any $v_i, v_j \in V$, there is a path from v_i to v_j. A connected graph without cycles is a *tree*, and a disjoint union of trees is called a *forest*.

A *rooted tree* is a tree T together with a designated vertex $r \in V(T)$ called the root. For a vertex $v \in V(T)$, we refer to the vertices on the unique path from r to v as the *ancestors* of v. If v_1 is an ancestor of v_2, we refer to v_2 as a *descendant* of v_1. We will denote the subtree of T induced by the set of all descendants of a vertex x in T as T_x. A disjoint union of rooted trees is a *rooted forest* F, and we refer to the roots of the trees as the roots of F. Note that this implies that every vertex is both ancestor and descendant to itself.

Hypergraph Decompositiosn A *tree decomposition* of hypergraph H is a tree T together with a function $\text{bag} : T \to 2^{V(H)}$ such that, for every edge $e \in E(H)$, there is a vertex $n \in V(T)$ such that $e \subseteq \text{bag}(n)$. Additionally, we require that the subgraph (of T) induced by $\{n \in V(T) : v \in \text{bag}(n)\}$ is connected for each $v \in V(H)$. We will refer to the set $\{\text{bag}(n) : n \in T\}$ as the bags of T. Throughout the paper we will consider tree decompositions to be rooted. Note that this is not a technical restriction as any arbitrary rooting is sufficient.

A *width function* f is a function that assigns a hypergraph H together with a subset $U \subseteq V(H)$ of its vertices a non-negative real number $f(H, U)$. We will commonly write this as $f_H(U)$, and, if the underlying hypergraph is clear from context, simply as $f(U)$. For a given tree decomposition (T, bag) of H, the f-width of (T, bag) is the maximal width of any bag, i.e. $\max_{v \in T} f(H, \text{bag}(v))$. The f-width of H is the minimal f-width over all tree decompositions of H. We denote this as $f\text{-width}(T)$ and $f\text{-width}(H)$ respectively. Now let H be a hypergraph and $U \subseteq V(H)$. We say a function $\lambda : E(H) \to \{0, 1\}$ is an edge cover of U if $\sum_{e: v \in e} \lambda(e) \geq 1$ for all $v \in U$. In other words, we choose a set of edges such that each vertex is included in at least one of them. The weight of an edge cover is $\sum_{e \in E} \lambda(e)$, and the edge cover number $\rho(H, U)$ is the minimal weight of an edge cover of U. The generalized hypertreewidth (ghw) of a hypergraph H is the $f\text{-width}(H)$ where $f = \rho$. The fractional cover number $\rho^*(H, U)$ is defined analogously, except that we can assign non-integer values to edges, i.e. $\lambda : E(H) \to [0, 1]$. The fractional hypertreewidth (fhw) of a hypergraph H is the $f\text{-width}(H)$ where $f = \rho^*$.

Factors and Elimination Forests Our technical development involves extensive combinatorial arguments about the structure of certain rooted forests. In particular we argue on the structure in terms of over so-called *factors* as introduced in [5].

Definition 3 (Factors)**.** For rooted forest T we define the following three kinds of factors:

Tree Factor A *tree factor* is a set $V(T_x)$ for some $x \in V(T)$. We call x *the root* of the factor.

Forest Factor A forest factor is a **nonempty** union of tree factors whose roots are siblings. In particular, a tree factor is also a forest factor but the empty set is not. The set of roots of the tree factors included in the union are called the *roots* of the factor. If they are not roots of components of T, we define their joint parent as the *parent* of the factor. Note that We

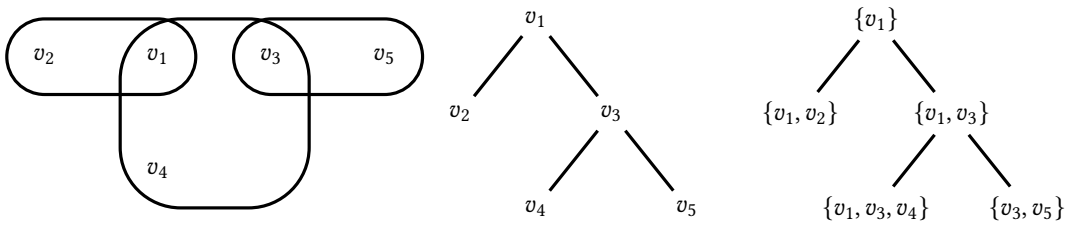


Fig. 1. An elimination forest for the depicted hypergraph (left) and its induced tree decomposition (right).

consider two nodes of T siblings when they have a joint parent or if they both roots of components of T. This convention is chosen to remain consistent with the notation of [5].

Context Factor A context factor is a set $U = X \setminus Y$ where X is a tree factor and Y is a forest factor whose roots are strict descendants of the root of X. The *root* of U, denoted by $rt(U)$, is the root of X and the roots of Y are called the *appendices* of U. Note that a tree factor is **not** a context factor. We will sometimes refer to X as the *base tree factor* of U and to Y as the *removed forest* of U. For clarity we sometimes denote X and Y as $X(U)$ and $Y(U)$. We also denote the parent of the appendices by $pr(U)$ and let the *spine* of U, denoted by $spine(U)$, be the path from $rt(U)$ to $pr(U)$

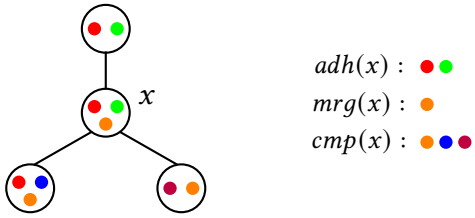
We say that two factors are *intersecting* if there have at least one vertex in common. Note also that a tree factor is also always forest factor.

Let X be a set of nodes of a forest T, an *X-factor* (wrt. T) is a factor F with $F \subseteq X$. A *factorization* of X is a partition of X into X-factors. A X-factor is maximal if no other X-factor contains it as a strict subset. Let $MF(X)$ denote the set of maximal X-factors. Note that by Lemma 3.2 [5] we have that $MF(X)$ is always a factorization of X.

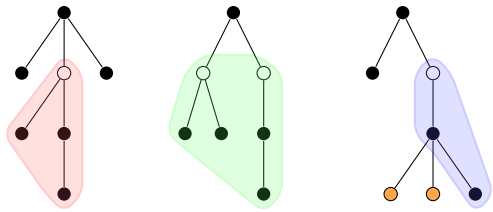
We will be interested in particular in factors of *elimination forests*. Intuitively, this is a tree-shaped arrangement of the vertices of a hypergraph that takes adjacency into account. We will see that this representation also naturally induces tree decompositions.

Definition 4 (Elimination Forest)**.** Let G be a hypergraph. An *elimination forest* T for G is a rooted tree over $V(G)$ such that $u, v \in e$ for some $e \in E(G)$ implies that either u is an ancestor of v in T or v is an ancestor of u. An elimination forest T induces a function $\text{bag}_T : V(T) \to 2^{V(G)}$ as follows: $\text{bag}_T(u)$ contains exactly u and all those ancestors v that are adjacent with u or a descendant of u. Note that (T, bag_T) is a tree decomposition of H. In what follows, when we refer to the f-width of an elimination forest, we mean the f-width of the tree decomposition induced by the forest.

Tree decompositions induced by elimination forests will play a central role in our technical developments. We will now fix some auxiliary definitions to simplify reasoning over tree decompositions. Note that while tree decompositions in this paper will generally be induced by an elimination forest, the definitions are independent of the forest and apply generally. To that end, for tree decomposition $\mathbf{t} = (T, B)$ we define the following. For a non-root node x of T with parent p, the *adhesion* is $adh(x) := \text{bag}(x) \cap \text{bag}(p)$. The adhesion of the root node is the empty set. The *margin* of a node x of T are those vertices of $\text{bag}(x)$ that are not in the adhesion of x, i.e., $mrg(x) := \text{bag}(x) \setminus adh(x)$. Finally, the *component* of node x of T is $cmp(x) := \bigcup_{y \in V(T_x)} mrg(y)$. The *target sets* of $\mathbf{t}$ are the components of nodes of T. That is $\mathsf{TS}(\mathbf{t}) = \{cmp(x) \mid x \in V(T)\}$. When there is an underlying elimination forest that is not clear from the description, we state it in the subscript: say $adh_T(x)$.



(a) The adhesion, margin and component of the node x.



(b) A Tree, Forest and Context Factor. Roots marked in white, appendices in orange.

Fig. 2. Illustrations of factors and key tree decomposition notions used in this paper.

MSO Transductions Ultimately our FPT algorithm will be stated in terms of MSO transductions. A transduction is a mapping of a model over the input signature to a set of models over the output signature that is defined as a composition of elementary transductions of Colouring, Copying, MSO Interpretation, Filtering, and Universe restriction. (a detailed description of elementary transductions is provided in [20], Appendix D). In particular, let $\mathbf{g}_1, \dots \mathbf{g}_p$ be a sequence of elementary transductions such that for each $2 \leq i \leq p$ the input signature of $\mathbf{g}_i$ is the same as the output signature of $\mathbf{g}_{i-1}$. Then the *transduction* $Tr[\mathbf{g}_1, \dots, \mathbf{g}_p]$ is a function over the models M of the input signature of $\mathbf{g}_1$ defined as follows. If $p = 1$ then $Tr[\mathbf{g}_1](M) = \mathbf{g}_1(M)$. Otherwise $Tr[\mathbf{g}_1, \dots, \mathbf{g}_p](M) = \bigcup_{M' \in \mathbf{g}_1(M)} Tr[\mathbf{g}_2, \dots, \mathbf{g}_p](M')$. The size of an MSO transduction is the sum of the sizes of its elementary MSO transductions, where colouring has size 1, copying has as size the number k of copies it produces, and for any other type of elementary transduction the size is the total size of MSO formulas (i.e., number of symbols) that make up its description. We write $\|\mathbf{Tr}\|$ for the size of transduction $\mathbf{Tr}$.

We note that in [5], a transduction is defined as a binary relation with pairs (M_L, M_R) where the 'left-hand' model M_L is over the input signature whereas the 'right-hand' model M_R is over the output signature. Our definition is equivalent as a mapping of a left-hand model to a set of right model determines a binary relation and vice versa. For a detailed description of MSO logic and formal languages we refer the reader to [8].

We will state MSO formulas over two specific signatures, τ_{td} for formulas over tree decompositions, and τ_{ef} for formulas over elimination forests. Formally, the signature τ_{td} consists of unary predicates *Vertex*, *Edge*, *Node* and binary predicates *Bag*, *Adjacenct*, *Descendant*. The signature τ_{ef} consists of unary predicates *Vertex*, *Edge* and binary relations *Child*, *Adjacenct*.

The intended meaning of these predicates is the following: for a model M over τ_{td}, *Vertex* and *Edge* are the vertices and (hyper)edges of a hypergraph $H(M)$, and *Adjacent* describes, for a given tuple (v, e), if vertex v is contained in the edge e. *Nodes* and *Descendant* describe the vertices and descendancy of a tree T and the predicate *Bag* induces a bag function *bag* (via $bag(t) = \{v \in Vertex \mid Bag(t, v)\}$) such that $\mathbf{t} = (T, bag)$ is a tree decomposition of $H(M)$. For a hypergraph H and a tree decomposition $\mathbf{t} = (T, \mathbf{B})$, we define the model $M = M(H, \mathbf{t})$ *induced* by H and $\mathbf{t}$ as the model M with $H = H(M)$ and $\mathbf{t}$ being the tree decomposition described by *Node*, *Descendant* and *Bag* relation of M.

For a model M over τ_{ef}, *Vertex*, *Edge* and *Adjacent* describe a hypergraph $H(M)$ (as with τ_{td}), while the predicates *Vertex* and *Child* model a directed graph $F(M)$ that is, in fact, will be proved to be an elimination forest of $H(M)$ if M belongs to the output of the transduction being constructed.

3 Technical Overview

To show our main result, we require a substantial amount of rather technical combinatorial arguments. In this section we therefore focus on providing a technical overview, presenting the main combinatorial results, how they are derived, and how they interact to form the final FPT algorithm. We first define the notion of a manageable width function in Section 3.1 In Section 3.2 and Section 3.3, we develop the main technical ingredients for the proof of the theorem and provide the proof in Section 3.4. Full proof details are provided in [20], Appendix sections A to D.

3.1 The Main Theorem

We isolate the minimal properties of a width function that we need in order to extend the transduction-based optimisation framework of Bojańczyk and Pilipczuk [5]. We call such width functions *manageable*. This class includes the edge-cover number ρ and its fractional relaxation ρ^*, and thus subsumes generalized and fractional hypertree width. At the same time, it is broad enough to plausibly capture further hypergraph width measures studied in the literature (e.g., variants based on modular width functions [21]).

A key obstacle compared to treewidth is that the relevant width functions are not compositional with respect to arbitrary partitions of a bag. We therefore distinguish a strong and a weak notion of separability.

Definition 5 (Monotone separability)**.** Let f be a width function. We say that f is *monotonically separable* if there exists a function $g_f : \mathbb{R}^2_{\geq 0} \to \mathbb{R}_{\geq 0}$ that is monotone in both arguments such that, for every hypergraph H, every $U \subseteq V(H)$, and every partition $U = U_1 \cup U_2$, we have

$$f_H(U) = g_f\big(f_H(U_1), f_H(U_2)\big).$$

We say that f is *weakly monotone separable* if the same condition holds for every *edge-disjoint* partition $U = U_1 \cup U_2$, i.e. no hyperedge of H intersects both U_1 and U_2. We refer to g_f as the *combiner* for f. Due to associativity, we will denote $g_F(g_F(n_1, n_2), n_3)$ as $g_F(n_1, n_2, n_3)$.

For instance, treewidth corresponds to $f_H(U) = |U|-1$ and is monotone separable with combiner $g_f(x, y) = x + y + 1$. In contrast, neither generalized nor fractional hypertree width is monotone separable: for $f = \rho$ (and similarly $f = \rho^*$), the value on U cannot be recovered from the values on an arbitrary partition $U_1 \cup U_2$. However, the monotone separability is restored under edge-disjoint partitions: if U_1 and U_2 are edge-disjoint, then $\rho_H(U) = \rho_H(U_1) + \rho_H(U_2)$ (and likewise for ρ^*), so these functions are weakly monotone separable.

The treewidth algorithm of [5] crucially relies on the underlying width function being monotone separable. Our main technical step is to replace it by weak separability, which requires substantial technical extension and additional structure to control how decompositions interact with edge-disjoint splits. We capture the extra requirements in the following definition.

Definition 6. We call a width function f *manageable* if it satisfies all of the following.

Weak monotone separability. f is weakly monotone separable, as in Definition 5.

Bounded size. There exists a computable function β such that for every $k \geq 0$, every hypergraph H, and every $U \subseteq V(H)$, if $f_H(U) \leq k$, then $|U| \leq \beta\big(k, \mathit{rank}(H)\big)$. We refer to any such β as a *size bound* for f.

Invariant locality. For all hypergraphs H_1, H_2 and all $U_1 \subseteq V(H_1)$, $U_2 \subseteq V(H_2)$, if $H_1[U_1] \cong H_2[U_2]$, then $f_{H_1}(U_1) = f_{H_2}(U_2)$.

Automatizability. There exists an algorithm that, given a hypergraph H, computes $f_H(V(H))$ in time at most $t(|V(H)|)$ for some computable function t.

Intuitively, invariant locality means that $f_H(U)$ depends only on the isomorphism type of the induced subhypergraph $H[U]$. Together with bounded size, this yields a *bounded number of values* phenomenon: for fixed k and r, there are only finitely many isomorphism types of induced subhypergraphs $H[U]$ with $rank(H) \leq r$ and $f_H(U) \leq k$, and hence only finitely many values $\leq k$ that f can take on such bags (bounded by a function of k and r). Finally, automatizability is necessary only at the final stage, to effectively enumerate (up to isomorphism) all small hypergraphs of f-width at most k.

We are now in a position to state our main result regarding exact computation of f-width in full formality.

Theorem 7 (Extended version of Theorem 1). *f-width-check is fixed-parameter tractable for every manageable width function f. In particular, there is an $g(k, rank(H), \Delta(H) \cdot |V(H)|)$, for some computable function g, time algorithm that takes as input a hypergraph H and a parameter k and returns a tree decomposition H with width f-width(H) if f-width$(H) \leq k$, or rejects. In the latter case it is guaranteed that f-width$(H) > k$.*

Recall, ghw is precisely f-width when f is the edge cover number ρ, and fhw is f-width when f is the fractional edge cover number ρ^*. It is trivial to verify that ρ is indeed manageable. For ρ^*, the bounded size follows from

$$|U| \leq \sum_{v \in U} \underbrace{\sum_{e : v \in e} \lambda(e)}_{\geq 1} = \sum_{e \in E} \sum_{v \in e} \lambda(e) \leq \sum_{e \in E} rank(H)\lambda(e) \leq k \cdot rank(H). \tag{1}$$

The other conditions of manageability are immediate.

The long-standing open question of fixed-parameter tractable algorithms for ghw and fhw then directly follow from Theorem 7.

Corollary 8. *For input hypergraph H and $k > 0$, it is fixed-parameter tractable to check* $\mathsf{ghw}(H) \leq k$ *and* $\mathsf{fhw}(H) \leq k$ *parameterised by $k, rank(H), \Delta(H)$.*

In the rest of this section, we will provide an overview of the proof of Theorem 7. Subsequent sections will then expand on the technical details of the statements made here.

3.2 The Structure of Optimal Elimination Forests

In this subsection, we develop the main combinatorial engine behind Theorem 7, which we call the *Reduced Hypergraph Dealternation Lemma*. The key to our algorithm is that we can check manageable f-width on elimination forests whose structure is simple. However, this simplicity is relative in the following sense. Let $\mathbf{t} = (T, \mathbf{B})$ be a tree decomposition of H whose f-width is not much larger than f-width(H). Then there exists an *optimal* elimination forest F for H (i.e., f-width$(F) = f$-width(H)) whose induced decomposition is *aligned* with $\mathbf{t}$. This alignment in particular refers to how factorisations of the elimination forest match up with $\mathbf{t}$. We will therefore be interested in measures that relate the factorisation of elimination forests to specific tree decompositions. In the following, we refer to a pair of a tree decomposition and an elimination forest for the same hypergraph H as a *TF-pair*.

We introduce two new measures to quantify the relevant structural alignment of *TF*-pairs. The first factor in such efficient computability is that we need each target set of $\mathbf{t}$ to have a small factorization in F. Intuitively, this restricts the possible ways to construct F from $\mathbf{t}$ (recall, target sets TS and maximal factors *MF* are defined in Section 2) .

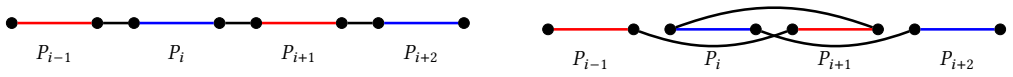


Fig. 3. A swap operation exchanging two consecutive monochromatic intervals.

Definition 9. For *TF*-pair $\mathbf{t}, F$, we define the *split* of the pair as the size of the largest maximal factorization of target sets of $\mathbf{t}$, i.e., $split(\mathbf{t}, F) := \max_{U \in \mathsf{TS}(\mathbf{t})} |MF(U)|$.

The second aspect to this structural alignment is that the individual factors themselves follow a kind of top-down monotonicity. Formally, this means that we want to avoid the presence of many context-factors in our representation. For the definition, it will be helpful to extend our notation. We call $U \subseteq V(H)$ *well-formed* w.r.t. F if $MF(U)$ contains no context factor (equivalently, it consists only of forest factors).

Definition 10. For *TF*-pair $\mathbf{t}, F$ we define the *irregularity* of a node x of $\mathbf{t}$ as the number of children y of x, such that $cmp(y)$ is not well-formed w.r.t. F. We denote the maximum irregularity of a node of T by $mir(\mathbf{t}, F)$.

These two measures of structural complexity, *split* and *mir*, are key to our framework. In key contrast to [5], we isolate split and irregularity as the only interface parameters necessary to build a modular, general theoretical framework for width checking using MSO transductions. It allows us to decouple various statements for treewidth into independent parts, connected only through split and irregularity. This abstraction and generalisation ultimately allows us to establish core properties also for width functions that are not monotone separable.

Lemma 11 (Reduced Hypergraph Dealternation). *Assume that f is manageable. There exists a monotone function γ such that for every hypergraph H and every tree decomposition $\mathbf{t}$ of H, there is an elimination forest F of H with f-width$(F) = f$-width(H) and*

$$split(\mathbf{t}, F),\ mir(\mathbf{t}, F)\ \leq\ \gamma\big(\mathrm{tw}(\mathbf{t}),\ f\text{-}width(H),\ rank(H),\ \Delta(H)\big).$$

Proof Sketch. We analyse partitions $(Red, X, Blue)$ of $V(H)$, where X separates *Red* from *Blue* (i.e., no hyperedge has both a red and a blue vertex). In the application, X is chosen as a subset of a bag B_x of $\mathbf{t}$, and hence $|X| \leq \mathrm{tw}(\mathbf{t}) + 1$. We further consider only *reduced* elimination forests: an elimination forest F is reduced if every parent–child edge (u, v) of F is witnessed by some hyperedge, i.e., there exists $e \in E(H)$ with $u \in e$ and $e \cap V(F_v) \neq \emptyset$, where F_v is the subtree rooted at v. We show that one can transform every elimination forest into a reduced one of the same width, hence this restriction can be made without loss of generality.

Fix a reduced optimal elimination forest F and a coloured context factor $U \subseteq Red \cup Blue$. Along the spine of U, consider the maximal monochromatic subpaths (intervals). The number of such intervals is a coarse measure of how often the spine alternates between the two sides of the cut, and thus of how "entangled" this factor is with the separator X: many intervals indicate a complicated interaction that we will later rule out in a well-aligned optimal forest. A *swap* exchanges two consecutive intervals along the spine. In the alternating pattern of four consecutive intervals, swapping the middle two makes adjacent intervals merge, reducing the total number of intervals by 2 while preserving connectivity and monochromaticity (cf. Figure 3).

The key claim is that sufficiently long alternation forces the existence of a *neutral* swap, i.e., a swap that does not increase f-width. To prove this, we classify intervals by their *interface* with the separator X. An X-*ball* expresses how the bags induced along an interval intersect X. Manageability implies that, for fixed $|X|$, f-width(H), $rank(H)$, and $\Delta(H)$, only boundedly many different X-balls

can occur (see [20], Section A.1). In particular, this follows from the *bounded size* property limiting the size of the interface, and *invariant locality* limiting the number of isomorphism types. Hence, if a spine contains many intervals, then many of them share the same X-ball. On such a homogeneous region, the effect of swapping depends only on bounded local information; a pigeonhole argument yields two consecutive intervals whose swap is neutral (see [20], Section A.2).

Finally, we choose among all reduced optimal forests one that minimizes an appropriate potential: first the total number of monochromatic intervals, and then the induced factorisation complexity. A neutral swap strictly decreases this potential, so a minimum contains no long alternation across any such separator. This bounded alternation translates into bounded maximal factorisations of target sets and bounded irregularity, giving the desired bounds on $split(\mathbf{t}, F)$ and $mir(\mathbf{t}, F)$. The full proof of the lemma is the content of [20], Appendix A. □

The combinatorial insight from Theorem 11 may have implications beyond the algorithm presented here. It is discussed in [5] that the Dealteration Lemma for treewidth in a sense reflects the combinatorial core that underlies the classic FPT algorithm for checking treewidth by Bodlaender and Kloks [4]. This is of particular note as no analogous algorithm is known for hypergraph width measures. For our main theorem we proceed similar to [5], by abstractly using the combinatorial result to express our problem in terms of bounded MSO transductions. However, Theorem 11 suggests that building on our introduction of *split* and *mir*, and their bounded character, may form the basis of future work into a more immediate (Bodlaender-Kloks style) FPT algorithm for deciding f-width.

3.3 Efficient Construction of Transductions

The second major technical challenge in proving Theorem 7 is the construction of a small transduction that relates tree decompositions to the optimal elimination forests of Theorem 11.

For a tree decomposition (T, bag) of a hypergraph H, mn is a function from $V(H)$ to $V(T)$ where u is mapped to the (unique) node x with $u \in mrg(x)$, we say x is the *the margin node of* u. We will primarily be interested in the paths in the tree decomposition induced by two margin nodes. For nodes u, v of T, let $Path_{mn}(u, v)$ be the set of vertices on the shortest path from $mn(u)$ to $mn(v)$ in T.

Definition 12. Let $\mathbf{t}, F$ be a TF-pair of H and let $u \in V(H)$. The *stain* of u (w.r.t. $\mathbf{t}, F$), denoted by $Stain(u, (\mathbf{t}, F))$, is the set consisting of $mn_{\mathbf{t}}(u)$ and the vertices of $Path_{mn}(u, v)$ for each child v of u in F. That is,

$$Stain(u, (\mathbf{t}, F)) = \{mn(u)\} \cup \bigcup_{v \in Child_F(u)} Path_{mn}(u, v)$$

Note that Definition 12 is different from the definitions of irregularity and split above in that it considers a part of F in the context of $\mathbf{t}$ rather than a part of $\mathbf{t}$ in the context of F. Intuitively, the stain of a vertex u represents subtree of $\mathbf{t}$ that interacts with the part F rooted at u. For the transduction it will important that the way these subtrees are coupled with each other is controlled. We capture this "coupling" more formally in terms of the *conflict graph* of a *TF*-pair $\mathbf{t}, F$, denoted by $\mathrm{CG}(\mathbf{t}, F)$, which has $V(H)$ as the set of vertices and u is adjacent to v if and only if their stains intersect. Analogous to [5], we show that there is an MSO transduction that relates $\mathbf{t}$ to F for every *TF* pair, assuming that the transduction was constructed with knowledge of the chromatic number of $\mathrm{CG}(\mathbf{t}, F)$.

Lemma 13. *There is an algorithm whose input is $q > 0$ and the output is a transduction* $\mathbf{Tr}$ *of size $O(q)$ whose input signature is τ_{td} and the output signature is τ_{ef}. Let $\mathbf{t}, F$ be a TF-pair of hypergraph*

H and let $M(H, \mathbf{t})$ be the relational structure over τ_{td} induced by H and $\mathbf{t}$. Then the following is true regarding $\mathbf{Tr}(M(H, \mathbf{t}))$.

(1) Let $M^* \in \mathbf{Tr}(M(H, \mathbf{t}))$. Then $H(M^*) = H$ and $F(M^*)$ is an elimination forest of H.

(2) If the chromatic number of $\mathrm{CG}(\mathbf{t}, F)$ is at most q, then there is a $M^* \in \mathbf{Tr}(M(H, \mathbf{t}))$ such that $F(M^*) = F$.

Fortunately, it is in fact possible to bound the chromatic number of CG in terms of the two central parameters of our framework, *split* and *mir*. Meaning in the context of Theorem 11, we can consider transductions from Theorem 13 in size bounded in terms of $\mathrm{tw}(\mathbf{t})$, $\mathbf{f}\text{-width}(\mathbf{H})$, $\mathit{rank}(\mathbf{H})$ and $\Delta(H)$.

Theorem 14. *There is a monotone function η such that the chromatic number of $\mathrm{CG}(\mathbf{t}, F)$ is at most $\eta(\mathit{split}(\mathbf{t}, F), \mathit{mir}(\mathbf{t}, F), \mathit{tw}(\mathbf{t}), \mathit{rank}(H))$.*

Proof Idea. The graph $\mathrm{CG}(\mathbf{t}, F)$ is an intersection graph of connected subgraphs of a forest, and therefore also chordal. Chordal graphs are perfect graphs, meaning their chromatic number is the size of their largest clique. We then use the fact that $\mathrm{CG}(\mathbf{t}, F)$ also enjoys the Helly property and further observations on the "density of stains" to ultimately bound the size of the largest clique in $\mathrm{CG}(\mathbf{t}, F)$. The full proof of the theorem is the content of [20], Appendix C. □

As a final step we need a further transduction that verifies that f-width of the forests produced by the transduction of Theorem 13. For treewidth, this can be done implicitly, as this corresponds to limiting the size of bags themselves. To lift the overall procedure to arbitrary manageable f-width, we add another step to the transduction that filters for those elimination forests with optimal f-width. We show that such a transduction indeed always exists, with the core properties encapsulated in the following technical lemma which is the main result of [20], Appendix B.

Lemma 15. *Let f be a manageable width function and let β be a size bound for f as in Definition 6. Then, for each $k \geq 0$ and positive integer r, the following statements hold.*

(i) There is an MSO sentence $\varphi^*_{f,k,r}$ over the signature τ_{ef} such that for a structure M over the signature, $M \models \varphi^*_{f,k,r}$ if and only if $F(M)$ is of f-width at most k.

(ii) There is an algorithm that outputs $\varphi^*_{f,k,r}$ with input k, r in time that depends only on a computable function in k, r and β.

(iii) Let $\mathcal{H}_{k,r}$ be all hypergraphs with $\mathit{rank}(H) \leq r$ and $f_H(V(H) \leq k$. Then there is an algorithm that computes the set $V = \{f_H(V(H)) \mid H \in \mathcal{H}_{k,r}\}$ in time depending on k, r, and β. In consequence, $|V|$ can also be bounded in terms of k, r, β.

3.4 Proof of Theorem 7

Putting together the contents of the previous two sections forms the spine of our algorithm. We will show that there is an MSO transduction that represents a sound and complete algorithm for computing an optimal width elimination forest from an initial tree decomposition. Using the following result of Bojańczyk and Pilipczuk and the bounds we obtain on the size of our transduction we then observe that the resulting algorithm is indeed fixed-parameter tractable.

Proposition 16 (Theorem 6.1 in [5]). *There is an algorithm that, given an MSO transduction $\mathbf{Tr}$ and a relational structure $\mathbb{A}$ over the signature of $\mathbf{Tr}$, implements $\mathbf{Tr}$ on $\mathbb{A}$ in time*

$$f(\|\mathbf{Tr}\|, \mathrm{tw}(\mathbb{A}))\ (\|\mathbb{A}\| + m),$$

where m is the size of the output structure (or 0 if empty) and f is computable.

Proof of Theorem 7. The algorithm first computes $p = \beta(k, rank(H))$ and runs an off-the-shelf algorithm for exact treewidth computation parameterized by $p - 1$. We note that, due to the bounded size property, if $f\text{-width}(H) \leq k$ then $\text{tw}(H) \leq p - 1$. Therefore, if the treewidth computation algorithm rejects, our algorithm also rejects.

Otherwise, we have a tree decomposition $\mathbf{t} = (T, B)$ of H with treewidth at most $p - 1$. We then compute $p_1 = \gamma(tw(\mathbf{t}), k, rank(H), \Delta(H))$ with γ as in Lemma 11, and $q = \eta(p_1, p_1, tw(\mathbf{t}), rank(H))$ with η as in Theorem 14. Finally, we apply the algorithm as in Lemma 13 with input q to create a transduction $\mathbf{Tr}$ as specified in the lemma.

Claim 17. *Assume that f-width$(H) \leq k$. Then there is $M^* \in \mathbf{Tr}(M(H, \mathbf{t}))$ such that f-width$(F(M^*)) = f$-width(H).*

Proof of Claim. By Lemma 11 and the monotonicity of function γ as in the lemma, there is an elimination forest F of H with $f\text{-width}(F) = f\text{-width}(H)$ and such that both $split(\mathbf{t}, F)$ and $mit(\mathbf{t}, F)$ are at most p_0. Next, by Theorem 14 and monotonicity of η, the chromatic number of $\text{CG}(\mathbf{t}, F)$ is at most q. It follows from Lemma 13 that $Tr(M(H, \mathbf{t}))$ contains a model M_1 with $F(M_1) = F$. ◁

Next, we apply the algorithm as in the last item of Lemma 15 to produce the set $Val_{f,k,r}$ of values as specified in Item (iii) of the lemma. Our algorithm explores the values $a \in Val_{f,k,r}$ in the increasing order. For each value a, a subprocedure tests whether $f\text{-width}(H) \leq a$ and either returns a corresponding tree decomposition or rejects. In the latter case it is guaranteed that $f\text{-width}(H) > a$. If the subprocedure rejects for all $a \in Val_{f,k,t}$, then by definition of $Val_{f,k,r}$ the overall algorithm is safe to reject too. Otherwise, let a be the first value for which a tree decomposition is returned. Once again, according to the properties of $Val_{f,k,r}$, $f\text{-width}(H) = a$ and hence the algorithm is safe to return the obtained tree decomposition. It thus remains to describe the subprocedure for testing $f\text{-width}(H) \leq a$ for a specific value $a \leq k$.

By Lemma 15, $\varphi^*_{f,a,r}$ (as in the description of the lemma) can be designed by an algorithm in at most $g_0(k, r)$ time. We extend the transduction $\mathbf{Tr}$ by appending one domain filtering transduction with $\varphi^*_{f,a,r}$ being the underlying MSO formula. We call the resulting transduction $\mathbf{Tr}^*_a$.

Claim 18. *The following two statements hold.*

(1) If f-width$(H) \leq a$ then $\mathbf{Tr}^*_a(M(H, \mathbf{t})) \neq \emptyset$. Moreover, for each $M^* \in \mathbf{Tr}^*_a(M(H, \mathbf{t}))$, $H(M^*) = H$ and the f-width of $F(M^*)$ is at most a.

(2) If f-width$(H) > a$ then $\mathbf{Tr}^*_a(M(H, \mathbf{t})) = \emptyset$.

Proof of Claim. Assume that $f\text{-width}(H) \leq a$. In particular, this means that $f\text{-width}(H) \leq k$. Hence, we apply Claim 17 and conclude that the output of $\mathbf{Tr}(M(H, \mathbf{t}))$ contains at least one model M^* as specified in the statement of this claim. This model satisfies the underlying MSO of the last elementary transduction of $\mathbf{Tr}^*_a(M(H, \mathbf{t}))$ and hence the latter is not empty. Let $M^* \in \mathbf{Tr}^*_a(M(H, \mathbf{t}))$. This means that $M^* \in \mathbf{Tr}(M(H, \mathbf{t}))$. Hence, by Lemma 13, $H(M^*) = H$ and $F(M^*)$ is an elimination forest for H. This means that M^* passes through the filtering of the last stage of $\mathbf{Tr}^*_a(M(H, \mathbf{t}))$ and thus $f\text{-width}(F) \leq a$. This proves the first statement of the claim.

For the second statement, assume that $f\text{-width}(H) > a$ and yet $\mathbf{Tr}^*(M(H, \mathbf{t})) \neq \emptyset$. Let $M^* \in \mathbf{Tr}^*_a(M(H, \mathbf{t}))$. This means that $M^* \in \mathbf{Tr}(M(H, \mathbf{t})$. By Lemma 13 and the assumption, $f\text{-width}(F(M^*)) > a$. This means that M^* does not pass through the filtering of the last step of $\mathbf{Tr}^*{}_a(M(H, \mathbf{t}))$ and hence cannot belong to the output. This contradiction proves the second statement of the claim. ◁

Having constructed $\mathbf{Tr}^*_a$, we feed it along with $M(H, \mathbf{t})$ to the input of the algorithm $\mathcal{A}$ as in Theorem 16. According to the theorem, $\mathcal{A}(\mathbf{Tr}^*{}_a, M(H, \mathbf{t}))$ returns an elimination forest of H

of width at most a or rejects and, in the latter case, it is guaranteed that $f\text{-width}(H) > a$. If an elimination forest returned it only remains to turn it into a tree decomposition of H induced by F that can be done in a polynomial time. It thus only remains to verify the FPT time bound for $\mathcal{A}(\mathbf{Tr}^*_a, M(H, \mathbf{t}))$.

The runtime of $\mathcal{A}(\mathbf{Tr}^*_a, M_0)$ is FPT parameterized by the size of $\mathbf{Tr}^*{}_a$ and $tw(M(H,t))$. By our initial choice of p, we have that $tw(M(H,t))$ is at most $\beta(k, rank(H))$. The size of $\mathbf{Tr}^*_a$ is the size of $\mathbf{Tr}$ plus the size of $\varphi^*_{f,a,r}$. The size of $\mathbf{Tr}$ is upper bounded by a function of q which itself is upper bounded by a function of $k, rank(H), \Delta(H)$ by definition. Finally, the length of the last step is upper bounded by a function of k and $rank(H)$ by Lemma 15. □

While our proof does not directly provide an easily implementable algorithm, it does finally confirm that FPT checking of ghw and fhw, and in fact computation of witnessing decompositions, is possible. We believe that the key insights for obtaining Theorem 7, as well as establishing that FPT algorithms are theoretically possible, can guide the development for practical algorithms in future work. While parameters $f\text{-width}(H)$ and $rank(H)$ are central to our approach, the degree is only necessary to obtain the bound in Theorem 11. Its role is an interesting topic for future work, and we discuss the matter in detail in Section 5.

4 Computation and Approximation of Adaptive Width Variants

Let $\mathcal{F}$ be a family of width functions. The $\mathcal{F}$-width of hypergraph H is defined as the $\sup_{f\in\mathcal{F}} f\text{-width}(H)$. Width measures based on such a whole class of width functions have become a central element of the study of the complexity of database query answering [21, 22]. Despite their importance, we are not aware of any algorithmic study of the complexity of deciding such width measures. Here we show how our framework for manageable width functions also applies to some of these cases.

Any function $f : V(H) \to \mathbb{R}_{\geq 0}$ naturally extends to a width function by setting $f(U) = \sum_{v\in U} f(v)$. Such width functions are called *modular*. Let us additionally require that f is a fractional independent set, i.e., $f(e) \leq 1$. Let $\mathcal{F}$ be the class of all such modular width function induced by fractional independent sets (for a given H). The $\mathcal{F}$-width is then the so-called *adaptive width* of H, a parameter that characterizes when deciding conjunctive queries is in FPT [22]. For simplicity we fix $\mathcal{F}$ to this class of modular width functions induced by fractional independent sets for the rest of this paper.

In this section, we discuss how algorithms for two restricted versions of adaptive width follow from the framework for FPT f-width checking described above.

Definition 19. Let H be a hypergraph and $\delta \geq rank(H)$ be an integer. We define $\mathcal{F}_\delta$ as the family of all modular width functions induced by fractional independent sets f where each $v \in V(H)$, $f(v) = \frac{a}{\delta}$ where $a \in \mathbb{N}_0$ and $0 \leq a \leq \delta$. Furthermore, define $\mathcal{F}^*_\delta$ as the set of all modular width functions induced by fractional independent sets f such that for each $v \in V(H)$ either $f(v) = 0$ or $f(v) \geq \frac{1}{\delta}$.

Thus, $\mathcal{F}_\delta$-width is the *discretized* version of adaptive width with bounded precision expressed by δ. $\mathcal{F}^*_\delta$-width relaxes this restriction further and only excludes functions that assign very small non zero weights. To apply our framework, first observe that any fixed $f \in \mathcal{F}_\delta$ satisfies the weak monotone separability and automatizability properties of Theorem 6. The bounded size property holds on the H^{f+}, the induced subhypergraph on vertices where $f(v) \neq 0$. Since $f\text{-width}(H) = f\text{-width}(H^{f+})$, simply operating on H^{f+} will be sufficient for us. The only missing part is invariant locality. Intuitively, we fix each $f \in \mathcal{F}_\delta$ as vertex labels on the hypergraph. On a technical level, this yields a labelled hypergraph (each node labelled with its value under f) for which f indeed

satisfies invariant locality. That is, at the technical level where we use our previous machinery on the width functions from $\mathcal{F}_\delta$, they are indeed manageable, and the machinery discussed in the previous section works without change.

The main result of this section is a reformulation of Theorem 2 as stated below.

Theorem 20. *There is an algorithm that takes as input a hypergraph H, a positive rational k and $\delta \in \mathbb{N}$ and decides if $\mathcal{F}_\delta$-width$(H) \leq k$. The runtime of this algorithm is FPT parameterized by $k, rank(H), \Delta(H), \delta$.*

We need to introduce some further notation. Let $\delta \in \mathbb{N}$ and let $\mathcal{U} = \{U_0, \ldots, U_\delta\}$ be a weak partition of $V(H)$. We write $f[U_0, \ldots, U_\delta]$ (or simply $f[\mathcal{U}]$) for the modular width function that, for every vertex v, assigns $f(v) = \frac{a}{\delta}$ when $v \in U_a$. For clarity we write $\hat{f}[\mathcal{U}]$ for the modular width function induced by the vertex weight function $f[\mathcal{U}]$.

The central statement for the proof of Theorem 20 is to is the following theorem. It follows from the same underlying machinery as Theorem 7, but instead of turning the width check transduction directly into an algorithm, we translate it to an MSO formula where a modular width function is given as input via second-order variables.

Theorem 21. *There is an algorithm $\mathcal{A}(H, \mathbf{t}, k, \delta)$, where H is a hypergraph, $\mathbf{t}$ is a tree decomposition of H, k is a non-negative rational, and $\delta \in \mathbb{N}$, that returns an MSO formula $\varphi(U_0, \ldots, U_\delta)$ where $\mathcal{U} = \{U_0, \ldots, U_\delta\}$ are free (monadic) second-order variables such that: $M \not\models \varphi(U_0, \ldots, U_\delta)$ exactly if $\mathcal{U}$ is a weak partition of $V(H)$, $f[\mathcal{U}]$ is a fractional independent set, and $\hat{f}[\mathcal{U}]$-width$(H) > k$.*

The runtime of the algorithm is FPT parameterized by the treewidth of $\mathbf{t}$, k, $rank(H)$, $\Delta(H)$ and δ and the size of the resulting formula is upper-bounded by a function depending on these parameters.

To prove Theorem 20 we will extend the formula from Theorem 21 by (indirect) second-order quantification over the possible width functions. In particular, with the notation as in Theorem 21, let $\psi = \forall U_0 \ldots \forall U_\delta \; \varphi(U_0, \ldots, U_\delta)$.

Claim 22. *$M \models \psi$ if and only if $\mathcal{F}_\delta$-width of H is at most k.*

Proof of Claim. Assume first that $M \models \psi$. For the sake of contradiction, assume that the $\mathcal{F}_\delta$-width of H is greater than k. This means that there is a function $f \in \mathcal{F}_\delta$ such that the f-width of H is greater than k. By definition of $\mathcal{F}_\delta$, there is a weak partition $U'_0 \ldots U'_\delta$ such that for each $i \in \{0, \ldots \delta\}$, U'_i consists of exactly those elements v such that $f(v) = \frac{i}{\delta}$. Clearly, $f = f[U'_0, \ldots U'_\delta]$ is a fractional independent set. By definition $M \not\models \varphi(U'_0, \ldots U'_\delta)$. Then $M \not\models \psi$ as witnessed by $U_0 = U'_0, \ldots, U_\delta = U'_\delta$ thus contradicting our assumption.

Assume now that $M \not\models \psi$. This means that there are $U_0 = U'_0, \ldots U_\delta = U'_\delta$ such that $M \not\models \varphi(U'_0 \ldots, U'_\delta)$. By definition, this means that $f = f[U'_0, \ldots, U'_\delta]$ is a fractional independent set of H and the f-width of H is greater than k. By definition of f, clearly $f \in \mathcal{F}_\delta$. It follows that $\mathcal{F}_\delta$-width of H is greater than k. ◁

We now sketch the design of the required algorithm. We observe first that $tw(H)$ is at most twice $\mathcal{F}_\delta$-width multiplied by $rank(H)$. The first step of the algorithm is running a linear time algorithm for treewidth computation parameterized by $2 \cdot k \cdot rank(H)$. If the treewidth construction algorithm rejects then our algorithm can safely return *False*. Otherwise, the treewidth construction algorithm returns a tree decomposition $\mathbf{t} = (T, \mathbf{B})$ of treewidth at most $k \cdot rank(H) \cdot \Delta(H)$. We run $\mathcal{A}(H, \mathbf{t}, k, \delta)$ as in the statement of Theorem 21. We note that, by definition of $\mathbf{t}$, the runtime of the algorithm $\mathcal{A}$ is FPT parameterized by $k, rank(H), \Delta(H)$, and δ. It follows that the size of $\varphi(U_0, \ldots, U_\delta)$ is upper bounded by such a function of these parameters. Clearly, that the runtime

of construction of ψ is FPT parameterized by the same parameters and the size of ψ can be upper bounded by a function dependent on $k, rank(H), \Delta(H)$, and δ.

By Claim 22, it only remains to test whether $M(H, \mathbf{t}) \models \psi$. This can be done within the required runtime according to Courcelle's theorem [8]. The full proof details for Theorem 20 are presented in [20], Appendix E.

As a corollary of Theorem 20 we can show that $\mathcal{F}^*_\delta$-width is FPT approximable with ratio 2 through a rounding argument that connects it to $\mathcal{F}_\delta$-width. We then use the algorithm of Theorem 20 with the width parameter being k.

Corollary 23. *There is an algorithm that takes as input a hypergraph H, a positive rational k and $\delta \in \mathbb{N}$. If the algorithm* accepts*, then $\mathcal{F}^*_\delta$-width$(H) \leq 2k$. If the algorithm* rejects*, then it is guaranteed that $\mathcal{F}^*_\delta$-width$(H) \geq k$. The runtime of this algorithm is FPT parameterized by $k, rank(H), \Delta(H), \delta$.*

It follows from Corollary 23, that if there is $\delta = \eta(k, rank(H), \Delta(H))$ such that for all H, $\mathcal{F}^*_\delta = \mathcal{F}$ then adaptive width has a ratio 2 FPT approximation parameterized by the width k, $rank(H)$, and $\Delta(H)$. Thus existence of such a function η is an interesting open question for future work.

5 Conclusion & Future Work

In this paper we have developed a general framework for fixed-parameter tractable computation of hypergraph width measures. By introducing the class of *manageable* width functions and as a special case we obtain the first exact FPT algorithms for deciding $\mathsf{ghw}(H) \leq k$ and $\mathsf{fhw}(H) \leq k$ under any non-trivial parameterisation. Our framework further yields an exact FPT algorithm for discretized adaptive width, as well as a factor-2 FPT approximation algorithm for the relaxed variant F^*_δ-width.

A natural question for further research is whether the parameter $\Delta(H)$ is necessary for fixed-parameter tractability of f-width computation. For the related discussion, we find it easier to confine ourselves to a specific parameter ghw. Then the open question is as follows: is testing $\mathsf{ghw}(H) \leq k$ fixed-parameter tractable if parameterized by k and $rank(H)$?

In order to upgrade the machinery of this paper to the parametrization by k and $rank(H)$ only, it is enough to get rid of $\Delta(H)$ in the statement of Lemma 31 in [20]. Formally this leads to the following question.

Open Question 1. Is there a function h such that the following holds. Let F be an elimination forest of H and let $(Red, X, Blue)$ be a red-blue partition of $V(H)$. Let B be a context factor of F and suppose that the number of monochromatic intervals of the spine of B is at least $h(|X|, \mathsf{ghw}(F), rank(H))$. Then B admits a neutral swap (cf.,[20], Lemma 31).

A positive answer to Question 1 will lead to an immediate upgrade of the algorithm in this paper to parameterization by k and $rank(H)$. In order to answer the question negatively, a counterexample of the following kind is needed. Pick $k, r > 2$ and an infinite set of hypergraphs $H_1, H_2, \ldots$, their respective elimination forests $F_1, F_2 \ldots$ and respective context factors $B_1, B_2, \ldots$ with the number of monochromatic intervals of these factors monotonically increasing and yet none of B_i admitting a neutral swap.

We conclude the section with an open question related to adaptive width.

Open Question 2. Are there a function δ and a constant $c \geq 1$ that for every hypergraph H, $\mathcal{F}^*_\delta$-width$(H) \leq adw(H)/c$, where $\delta = \delta(k, rank(H), \Delta(H))$?

A positive answer to Question 2, combined with the results of the previous section will imply a constant ratio – in particular ratio $2c$ – FPT approximation algorithm for adaptive width

parameterized by the width, the rank, and the max-degree. Once again, the question is of a purely combinatorial nature.

References

[1] Albert Atserias, Martin Grohe, and Dániel Marx. 2013. Size Bounds and Query Plans for Relational Joins. *SIAM J. Comput.* 42, 4 (2013), 1737–1767. doi:10.1137/110859440

[2] Hans L. Bodlaender. 1996. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM J. Comput.* 25, 6 (1996), 1305–1317. doi:10.1137/S0097539793251219

[3] Hans L. Bodlaender and Ton Kloks. 1996. Efficient and Constructive Algorithms for the Pathwidth and Treewidth of Graphs. *J. Algorithms* 21, 2 (1996), 358–402. doi:10.1006/JAGM.1996.0049

[4] Hans L Bodlaender and Ton Kloks. 1996. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms* 21, 2 (1996), 358–402.

[5] Mikolaj Bojańczyk and Michal Pilipczuk. 2022. Optimizing tree decompositions in MSO. *Log. Methods Comput. Sci.* 18, 1 (2022). doi:10.46298/LMCS-18(1:26)2022

[6] Marco Bressan, Julian Brinkmann, Holger Dell, Marc Roth, and Philip Wellnitz. 2025. The Complexity of Counting Small Sub-Hypergraphs. *arXiv preprint arXiv:2506.14081* (2025).

[7] Marco Bressan, Matthias Lanzinger, and Marc Roth. 2023. The Complexity of Pattern Counting in Directed Graphs, Parameterised by the Outdegree. In *Proceedings STOC 2023*. ACM, 542–552. doi:10.1145/3564246.3585204

[8] Bruno Courcelle and Joost Engelfriet. 2012. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*. Encyclopedia of mathematics and its applications, Vol. 138. Cambridge University Press. http://www.cambridge.org/fr/knowledge/isbn/item5758776/

[9] Georg Gottlob and Gianluigi Greco. 2013. Decomposing combinatorial auctions and set packing problems. *J. ACM* 60, 4 (2013), 24:1–24:39. doi:10.1145/2508028.2505987

[10] Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. 2005. Pure Nash Equilibria: Hard and Easy Games. *J. Artif. Intell. Res.* 24 (2005), 357–406. doi:10.1613/JAIR.1683

[11] Georg Gottlob, Martin Grohe, Nysret Musliu, Marko Samer, and Francesco Scarcello. 2005. Hypertree Decompositions: Structure, Algorithms, and Applications. In *Proceedings of WG 2005 (Lecture Notes in Computer Science, Vol. 3787)*. Springer, 1–15. doi:10.1007/11604686_1

[12] Georg Gottlob, Matthias Lanzinger, Reinhard Pichler, and Igor Razgon. 2020. Fractional Covers of Hypergraphs with Bounded Multi-Intersection. In *Proceedings of MFCS 2020 (LIPIcs, Vol. 170)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 41:1–41:14. doi:10.4230/LIPICS.MFCS.2020.41

[13] Georg Gottlob, Matthias Lanzinger, Reinhard Pichler, and Igor Razgon. 2021. Complexity Analysis of Generalized and Fractional Hypertree Decompositions. *J. ACM* 68, 5, Article 38 (Sept. 2021), 50 pages. doi:10.1145/3457374

[14] Georg Gottlob, Nicola Leone, and Francesco Scarcello. 2002. Hypertree Decompositions and Tractable Queries. *J. Comput. Syst. Sci.* 64, 3 (2002), 579–627. doi:10.1006/JCSS.2001.1809

[15] Georg Gottlob, Zoltan Miklos, and Thomas Schwentick. 2007. Generalized hypertree decompositions: np-hardness and tractable variants. In *Proceedings of PODS 2007*. Association for Computing Machinery, New York, NY, USA, 13–22. doi:10.1145/1265530.1265533

[16] Martin Grohe and Dániel Marx. 2014. Constraint Solving via Fractional Edge Covers. *ACM Trans. Algorithms* 11, 1, Article 4 (Aug. 2014), 20 pages. doi:10.1145/2636918

[17] Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. 2019. On the Parameterized Complexity of Approximating Dominating Set. *J. ACM* 66, 5 (2019), 33:1–33:38. doi:10.1145/3325116

[18] Viktoriia Korchemna, Daniel Lokshtanov, Saket Saurabh, Vaishali Surianarayanan, and Jie Xue. 2024. Efficient Approximation of Fractional Hypertree Width. In *Proceedings of FOCS 2024*. IEEE, 754–779. doi:10.1109/FOCS61266.2024.00053

[19] Matthias Lanzinger and Igor Razgon. 2024. FPT Approximation of Generalised Hypertree Width for Bounded Intersection Hypergraphs. In *Proceedings of STACS 2024 (LIPIcs, Vol. 289)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 48:1–48:17. doi:10.4230/LIPICS.STACS.2024.48

[20] Matthias Lanzinger, Igor Razgon, and Daniel Unterberger. 2025. FPT Parameterisations of Fractional and Generalised Hypertree Width. *CoRR* abs/2507.11080 (2025). arXiv:2507.11080 doi:10.48550/ARXIV.2507.11080

[21] Dániel Marx. 2011. Tractable Structures for Constraint Satisfaction with Truth Tables. *Theory Comput. Syst.* 48, 3 (2011), 444–464. doi:10.1007/S00224-009-9248-9

[22] Dániel Marx. 2013. Tractable Hypergraph Properties for Constraint Satisfaction and Conjunctive Queries. *J. ACM* 60, 6 (2013), 42:1–42:51. doi:10.1145/2535926

[23] Hung Q Ngo, Ely Porat, Christopher Ré, and Atri Rudra. 2018. Worst-case optimal join algorithms. *J. ACM* 65, 3 (2018), 1–40.

[24] Neil Robertson and P.D Seymour. 1986. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms* 7, 3 (1986), 309–322. doi:10.1016/0196-6774(86)90023-4

Received December 2025; accepted February 2026