

WIDTH FUNCTIONS
FOR
HYPERTREE DECOMPOSITIONS

Isolde Adler

Dissertation zur Erlangung des Doktorgrades
der Fakultät für Mathematik und Physik
der Albert-Ludwigs-Universität Freiburg im Breisgau
January 2006

Dekan:	Prof. Dr. Josef Honerkamp
Gutachter:	Prof. Dr. Jörg Flum
	Prof. Dr. Martin Grohe
Datum der mündl. Prüfung:	18.04.2006

Contents

Prologue	1
1 Tree-width	3
1.1 Tree decompositions of infinite graphs	3
1.2 Simplicial decompositions	6
1.3 Tree-width and k -trees	9
1.3.1 Smooth tree decompositions	9
1.3.2 k -trees	10
2 f-hypertree-width of graphs	14
2.1 Definitions and some observations	15
2.1.1 Hypertree decompositions and width functions	15
2.1.2 Application to hypergraphs	17
2.1.3 Monotone width functions	19
2.2 A game for f -hypertree-width	22
2.2.1 Robber and cops	22
2.2.2 Strategies	24
2.2.3 Monotone strategies	26
2.3 The homomorphism problem	30
2.3.1 Structures with partial functions	31
2.3.2 A general no-promise algorithm	32
2.3.3 A general promise algorithm	36
2.3.4 Conjunctive query evaluation	39
2.4 Related f -invariants	41
2.4.1 Conditions on width functions	42
2.4.2 Obstructions: Brambles and tangles	44
2.4.3 Branch decompositions	46
2.4.4 Linking all invariants together	49
3 Hypergraphs	53
3.1 Definitions and some observations	54
3.1.1 Hypergraph pairs	54
3.1.2 Properties of c_H	56
3.1.3 c_H -hypertree-width is smaller than tree-width	57
3.2 The relation of c_H -cw and c_H -ghw	59
3.2.1 Hypergraph pairs	59
3.2.2 Implementable hypergraph pairs	62
3.2.3 Hypergraphs	64
3.3 The relation of c_H -ghw and c_H -hw	64
3.3.1 Hypergraphs	65
3.3.2 Ultramonotonicity	67
3.3.3 Hypergraph pairs	71

3.4	The hypergraph sandwich problem	72
3.4.1	Acyclicity	72
3.4.2	Hypergraph pairs and the HSP	73
4	Compactness	76
4.1	Triangulations	76
4.2	Compactness of f -ghw	77
4.3	Compactness of c_H -ghw	79
4.4	Non-compactness of c_H -hw	80
4.4.1	Hypergraph pairs	80
4.4.2	Hypergraphs	81
4.5	k -decomposability and f -ghw	84
5	More width functions	86
5.1	Mixed graphs	87
5.1.1	Definitions and some observations	87
5.1.2	Translation between hypergraphs and mixed graphs	90
5.2	Directed hypergraphs	94
5.2.1	Definitions and some observations	94
5.2.2	More on the homomorphism problem for non-relational signatures	99
5.3	Fractional edge covers	100
5.3.1	Definitions and some observations	101
5.3.2	Properties of fc_H	104
5.3.3	Fractional edge covers and directed hypergraphs	106
	Epilogue	108
	References	110
	Index	113

Prologue

Tree-width of graphs (cf. Chapter 1) is a fundamental concept in graph structure theory, and it also has important algorithmic applications. The tree-width of a graph measures how close it is to being a tree. Many problems that are NP complete in general become tractable when restricted to instances of bounded tree-width.

The analogous concept for hypergraphs is generalised hypertree-width, which can be defined as follows. A hypergraph $H = (V, E)$ (i. e. E is a set of finite subsets of the vertex set V , called the hyperedges of H) has generalised hypertree-width at most k if there is a tree T and a family $(B_t)_{t \in T}$ of pieces $B_t \subseteq V$ such that:

1. For all hyperedges $h \in E$ there is a tree node $t \in T$ such that $h \subseteq B_t$.
2. For all $v \in V$ the set $\{t \in T \mid v \in B_t\}$ is non-empty and connected in T .
3. For all tree nodes $t \in T$ there are hyperedges $h_1, \dots, h_k \in E$ such that $B_t \subseteq h_1 \cup \dots \cup h_k$.

There are many other definitions which are either equal to generalised hypertree-width or approximate it within a factor of 3. Again, many NP complete problems become tractable when restricted to instances of bounded generalised hypertree-width (cf. Chapters 2 and 3).

Generalised hypertree-width is ‘compact’ in the following sense. For a large class of infinite hypergraphs the generalised hypertree-width is the supremum of the generalised hypertree widths of the finite induced subhypergraphs (cf. Chapter 4).

The tree-width of a hypergraph is the tree-width of its underlying graph. A class of hypergraphs of bounded tree-width also has bounded generalised hypertree-width, but the converse is false. Hence generalised hypertree-width is better for algorithmic applications in so far as it gives rise to larger tractable classes. This is not the end of the story. M. Grohe and D. Marx obtained even larger classes by bounding the fractional hypertree-width [GM05]. A new approach is to consider directed hypergraphs and make use of the added information. (Both approaches are presented in Chapter 5.)

Tree-width, generalised hypertree-width and the generalisations from Chapter 5 can all be treated in the following framework. Let $G = (V, E)$ be a graph, and let f be a width function on G , i. e. a function assigning to every finite subset $X \subseteq V$ a real number $f(X)$ or $f(X) = \infty$. Then $f\text{-ghw}(G) \leq k$ if there are T and $(B_t)_{t \in T}$ satisfying conditions 1-2 above and

- 3'. For all tree nodes $t \in T$ there is a set $X \supseteq B_t$ such that $f(X) \leq k$.

If we choose $f(X) = |X| - 1$, then $f\text{-ghw}(G)$ is the tree-width of G , and for $f(X) = \min\{k \mid X = h_1 \cup \dots \cup h_k; h_i \in E\}$ and G the underlying graph of H it turns out that $f\text{-ghw}(G)$ is the generalised hypertree-width of H .

The following are the main contributions of this thesis:

- The framework for $f\text{-ghw}(G)$.

- Application of the framework to directed hypergraphs. E.g. this allows us to obtain larger tractable classes of conjunctive queries on databases if there are functional dependencies.
- Compactness of generalised hypertree-width (and in fact $f\text{-ghw}(G)$), under a reasonable condition.
- Generalisation of some well-known results to the infinite case, e.g. the characterisation of generalised hypertree-width of an infinite hypergraph in terms of a monotone robber and cops game.
- Generalised hypertree-width is within a constant factor of several other invariants which resemble graph invariants such as bramble-number, tangle-number, branch-width and non-monotone cop-width. This is essentially the result of joint work with G. Gottlob and M. Grohe [AGG05], although the translation to the $f\text{-ghw}$ framework necessitated some changes.
- Numerous complicated examples showing that various desirable properties do *not* hold.

Acknowledgements

I wish to thank everyone who supported me with my work. Particularly, I thank my supervisor Prof. Dr. Jörg Flum for his support, advice, confidence and encouragement during my work on this thesis. I am also very grateful to Prof. Dr. Martin Grohe for fruitful discussions during and after a week of joint work in Edinburgh, and to Hans Adler, Moritz Müller, and Mark Weyer for discussions and comments on drafts of this thesis or talks I gave in our research training group. Moreover, I thank the Deutsche Forschungsgemeinschaft for making this thesis possible.

Very special thanks to my daughter Sofia for enriching my life since almost fifteen months, and my mother Brigitte Adler for lovingly taking care of Sofia for more than two months time during the final spurt. Many thanks to all my dear friends for accompanying me through sunshine, rain and snow.

Chapter 1

Tree-width

There are numerous equivalent definitions for the tree-width $\text{tw}(G)$ of a graph G , which measures how similar G is to a tree.

the three historically most important definitions and show that they all agree. Tree-width was independently developed by three groups (see also [Di97, Chapter 12], for historical remarks):

- N. Robertson and P. D. Seymour [RS86a].
 $\text{tw}(G)$ is the least integer k such that G has a tree decomposition such that every block has at most $k + 1$ elements (Section 1.1).
- R. Halin [Ha64, Ha76].
 $\text{tw}(G)$ is the least integer k such that G has a triangulation which has no infinite $(k + 2)$ -clique as a subgraph (Section 1.2).
- D. Rose, S. Arnborg, A. Proskurowski et al. [Ro70, BP71, Ro74, AP81].
 $\text{tw}(G)$ is the least integer k such that G is a (partial) subgraph of a k -tree (Section 1.3).

In the first section we fix notation and present the definition of tree-width in the style given by Robertson and Seymour in their graph minor project. This will be the base for all later chapters. In the second section we use Halin's deep theorem on simplicial decompositions. We will need this section only in Chapter 4. Finally, in the last section of this chapter we present the third approach in terms of k -trees (which we will not need later).

1.1 Tree decompositions of infinite graphs

In this section we define the tree-width of a graph and we prove some basic results that we will need later on. We begin by fixing notation. We will generally, though not in every detail, follow [Di97].

For a set S we use the notation $\mathcal{P}(S) := \{S_0 \mid S_0 \subseteq S\}$, for an integer $k \geq 0$ we set $\mathcal{P}_{<k}(S) := \{S_0 \mid S_0 \subseteq S, |S_0| < k\}$, $\mathcal{P}_{=k}(S) := \{S_0 \mid S_0 \subseteq S, |S_0| = k\}$, and $\mathcal{P}_{<\omega}(S) := \{S_0 \mid S_0 \subseteq S, S_0 \text{ finite}\}$.

Definition 1.1.1

- A graph is a pair $G = (V(G), E(G))$ where $V(G)$ is a (possibly infinite) nonempty set of vertices, and $E(G) \subseteq \mathcal{P}_{=2}(V(G))$ is the set of edges of G . (Thus the edges of G are two-element sets of vertices.)

- G_0 is a (partial) subgraph of a graph G , $G_0 \subseteq G$, if $V(G_0) \subseteq V(G)$ and $E(G_0) \subseteq E(G)$. A subgraph $G_0 \subseteq G$ is an induced subgraph, if $E(G_0) = E(G) \cap \mathcal{P}_{=2}(V(G_0))$. For $X \subseteq V(G)$, the induced subgraph of G with vertex set X is denoted by $G[X]$. $G \setminus X$ denotes the induced subgraph $G[V(G) \setminus X]$. The union $G_1 \cup G_2$ of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the graph $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. The intersection $G_1 \cap G_2$ of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the graph $G_1 \cap G_2 = (V_1 \cap V_2, E_1 \cap E_2)$.
- The set $X \subseteq V(G)$ is connected if $G[X]$ is a connected graph. $C \subseteq V(G)$ is a connected component of G if $G[C]$ is a maximal connected subgraph of G .
- A forest is a graph without cycles. A tree is a connected forest. For a tree T we often write $t \in T$ instead of $t \in V(T)$ for the tree nodes. A branch D of T is the vertex set $D \subseteq V(T)$ of a maximal path in T that begins with the root.
- A directed tree T is a rooted tree where the edges are directed away from the root. We write (t, s) for an edge of T directed from the tree node t to the tree node s . If $(t, s) \in E(T)$, we say that the node s is a successor of the node t , and t is the (unique) predecessor of s , in symbols: $t = \text{pred}(s)$.
- A graph K is a clique (also complete graph), if $E(K) = \mathcal{P}_{=2}(V(K))$. A k -clique is a complete graph with k vertices.

Definition 1.1.2 Let G be a graph. A tree decomposition of G is a pair (T, B) , consisting of a tree¹ T and a family $B = (B_t)_{t \in T}$ of finite subsets of $V(G)$, the pieces of T , satisfying:

- (TD1) For each $v \in V(G)$ there exists $t \in T$, such that $v \in B_t$. We say the node t covers v .
- (TD2) For each edge $e \in E(G)$ there exists $t \in T$, such that $e \subseteq B_t$. We say the node t covers e .
- (TD3) For each $v \in V(G)$ the set $\{t \in T \mid v \in B_t\}$ is connected in T .

The width of a tree decomposition of G is defined as

$$\text{width}(T, B) := \sup \{ -1 + |B_t| \mid t \in T \} \in \omega \cup \{\infty\}.$$

The tree-width of G is defined as

$$\text{tw}(G) := \min \{ \text{width}(T, B) \mid (T, B) \text{ is a tree decomposition of } G \} \in \omega \cup \{\infty\}.$$

Note that if G has no isolated vertices, (i.e. every vertex $v \in V(G)$ is contained in some edge of G), then (TD2) implies (TD1). It is not hard to see that $\text{tw}(G) = 0$ if, and only if, G has no edges, and $\text{tw}(G) \leq 1$ iff G is a forest (cf. Example 1.1.5, 1, below).

Remark 1.1.3 Let G be a graph, and let $X \subseteq V(G)$ be a connected subset of G . Then the set $\{t \in T \mid X \cap B_t \neq \emptyset\}$ is connected in T .

Proof. For every $v \in X$ the set $\{t \in T \mid v \in B_t\}$ is connected in T by (TD3). Moreover, by (TD2) every edge $\{u, v\} \in G[X]$ is covered by some $t_{\{u, v\}} \in T$. Thus $t_{\{u, v\}} \in (\{t \in T \mid u \in B_t\} \cap \{t \in T \mid v \in B_t\}) \neq \emptyset$. Therefore, since X is connected, the set $\bigcup_{v \in X} \{t \in T \mid v \in B_t\} = \{t \in T \mid X \cap B_t \neq \emptyset\}$ is also connected. $\square$

¹Sometimes it is more convenient to work with a directed tree T . In this case we choose a root $r \in T$ and direct the edges of T away from the root.

Remark 1.1.4 *The class of all graphs G with $\text{tw}(G) \leq k$ is closed under taking subgraphs.*

Proof. Let (T, B) be a tree decomposition of G of width $\leq k$ and let $G' \subseteq G$ be a subgraph. Set $B'_t := B_t \cap V(G')$. It is easy to see that (T, B') is a tree decomposition of G' of width $\leq k$. $\square$

The remark stays true if we replace ‘subgraphs’ by ‘minors’. But we will not be concerned with minors.

Example 1.1.5

1. If G is a tree, then $\text{tw}(G) = 1$. This is witnessed by the following tree decomposition (T, B) :

$$\begin{aligned} V(T) &= V(G) \cup E(G), \\ E(T) &= \{\{v, e\} \mid v \in e \in E(G)\}, \\ B_v &= \{v\}, \text{ and } B_e = e. \end{aligned}$$

2. The countably infinite clique $K_{\aleph_0} = (\aleph_0, \mathcal{P}_{=2}(\aleph_0))$ satisfies $\text{tw}(K_{\aleph_0}) = \infty$. Moreover, $K_{\aleph_0}$ has a tree decomposition of infinite width: An infinite path T with an increasing sequence of finite pieces $B_t \subseteq V(K_{\aleph_0})$.
3. From 2 together with Remark 1.1.4 it follows that the uncountable clique $K_{\aleph_1} = (\aleph_1, \mathcal{P}_{=2}(\aleph_1))$ also has $\text{tw}(K_{\aleph_1}) = \infty$. The reader is invited to show that in fact $K_{\aleph_1}$ has no tree decomposition (hint: Proposition 4.3.1, 2), and thus $\text{tw}(K_{\aleph_1}) = \min\{\} = \infty$.

For tree decompositions of infinite graphs as in the example we need to admit infinite trees. We do so even for finite graphs. Fortunately, this does not affect tree-width of finite graphs:

Proposition 1.1.6 *Let G be a finite graph and let (T, B) be a tree decomposition of G . Then G has a tree decomposition (T', B') such that*

1. T' is finite,
2. for every node $t' \in T'$ there exists a node $t \in T$ such that $B'_{t'} = B_t$, and
3. in particular $\text{width}(T', B') \leq \text{width}(T, B)$.

Proof. Given (T, B) , choose a root $r \in T$ and direct the edges away from the root. Intuitively, for every $t \in T$ simultaneously we now contract the successors s of t into t that satisfy $B_s \subseteq B_t$:

We say that $s_1, s_2 \in T$ are equivalent, $s_1 \sim s_2$, if s_1 and s_2 have a common ancestor $t \in T$, such that the path $P^i = t, t_i^1, t_i^2, t_i^3, \dots, s_i$ from t to s_i (for $i = 1, 2$) satisfies $B_t \supseteq B_{t_i^1} \supseteq B_{t_i^2} \supseteq B_{t_i^3} \supseteq \dots \supseteq B_{s_i}$. It is straightforward to see that $\sim$ is an equivalence relation.

Now define (T', B') by $V(T') := T / \sim$, and

$$E(T') := \{(s'_1, s'_2) \mid \text{there are nodes } s_1 \in s'_1 \text{ and } s_2 \in s'_2 \text{ such that } (s_1, s_2) \in E(T)\}.$$

It is not hard to see that T' is a directed tree. Define $B'_{t'} := \bigcup_{t \in t'} B_t$. It is easy to see that (T', B') is a tree decomposition.

1: For every directed edge $(t', s') \in E(T')$ there is a graph vertex $v \in V(G)$ such that $v \in B_{s'} \setminus B_{t'}$. Thus each node $t' \in T'$ has at most $|V(G)|$ successors.

For the same reason, every branch of T' has at most length $|V(G)|$, and hence T' is finite. 2: Since for every equivalence class $t' \in V(T)$ we have a node $t \in t'$ that is a common ancestor of all $s \in t'$, we have $B'_{t'} = B_t$. 3: follows from 2. $\square$

Definition 1.1.7

- For a directed tree T and a tree node $t \in T$, let T_t denote the complete subtree of T with root t , i. e. the subtree containing t and all nodes reachable from t .
- For a tree decomposition (T, B) , where T is a directed tree, we set $B_{T_t} := \bigcup_{s \in V(T_t)} B_s$.

Proposition 1.1.8 *Let G be a non-empty graph, $K \subseteq G$ a finite clique in G , and let (T, B) be a tree decomposition of G . Then there exists a node $t \in T$ with $V(K) \subseteq B_t$.*

Proof. Let (T, B) be a tree decomposition of G . Starting with the root r of T we find a path leading us to a node $t \in T$ with $V(K) \subseteq B_t$ as follows:

Set $p_0 := r$. If $V(K) \subseteq B_r$ we are finished. Otherwise, connectedness (TD3) shows that for each $v \in V(K) \setminus B_r$ there exists a unique successor s_v of r such that $v \in B_{T_{s_v}}$. If $u \in V(K) \setminus B_r$ and $u \neq v$, the edge $\{u, v\}$ must be covered by some $s' \in T$. Connectedness (TD3) implies that $s' \in V(T_s)$ and hence $s_u = s_v$. We set $p_1 := s_v$. By (TD3) we have $(B_{p_0} \cap V(K)) \subseteq (B_{p_1} \cap V(K))$.

Suppose the path $p_0, p_1, \dots, p_n$ is already defined. If $V(K) \subseteq B_{p_n}$ we are finished. Otherwise, by the same arguments as above, there exists a unique successor s of p_n with $V(K) \setminus B_{p_n} \subseteq B_{T_s}$. We set $p_{n+1} := s$. By (TD3) we have $(B_{p_n} \cap V(K)) \subseteq (B_{p_{n+1}} \cap V(K))$.

Since $V(K)$ is finite, after finitely many steps the path ends at a node p_m with $V(K) \subseteq B_{p_m}$. $\square$

1.2 Simplicial decompositions

In this section we will prove a theorem on tree decompositions of infinite graphs by means of Halin's deep theorem on simplicial decompositions. We begin with the necessary definitions, followed by the statement of the fact which we are going to generalise to infinite graphs. This section is the basis for the main result of Chapter 4, but it will not be used otherwise. Readers who wish to skip the results on infinite graphs (and infinite hypergraphs), may continue with Chapter 2.

Definition 1.2.1

- A graph edge which joins two vertices of a cycle but is not itself an edge of the cycle is called a chord.
- A graph is chordal if it contains no induced cycle of length ≥ 4 . (Or, equivalently, if every cycle of length ≥ 4 has a chord.)
- A graph G' is a triangulation of the graph G , if G' is chordal, $V(G) = V(G')$ and $E(G) \subseteq E(G')$.

The following fact is [Di97, Proposition 12.3.11].

Fact 1.2.2 *Let G be a finite graph. G is chordal if and only if G has a tree decomposition (T, B) into complete pieces B_t (i.e. for all $t \in T$, the piece B_t induces a clique in G).*

From this it is easy to derive the fact that even infinite graphs that have a tree decomposition into complete pieces are chordal. This will be done below in Theorem 1.2.5, 1. The (much harder) converse was proved by I. Kříž and R. Thomas in [KT91] for infinite chordal graphs that have a finite upper bound on the sizes of complete subgraphs. They noted: ‘The following result can be deduced from Halin’s theory of simplicial decompositions, but we prove it from first principles’. We will follow the first path here because we need a slightly more general result.

Here are the main definitions and the main result of [Ha64]:

Definition 1.2.3 *Let G be a graph.*

- G is prime, if G is nonempty and connected, and G contains no clique K such that $G \setminus K$ is disconnected. We say G contains no separating clique.
- Let δ be an ordinal and $(G_\alpha)_{\alpha < \delta}$ a family of subgraphs of G . The family $(G_\alpha)_{\alpha < \delta}$ is called a simplicial decomposition of G , if it satisfies:

$$(SD1) \quad G = \bigcup_{\alpha < \delta} G_\alpha,$$

$$(SD2) \quad K_\beta = G_\beta \cap \bigcup_{\alpha < \beta} G_\alpha \text{ is a clique for every } \beta < \delta,$$

$$(SD3) \quad G_\beta \supsetneq K_\beta \text{ and } \bigcup_{\alpha < \beta} G_\alpha \supsetneq K_\beta.$$

It follows from (SD1) and (SD2) that the subgraphs G_α are actually induced subgraphs.

Fact 1.2.4 (Halin 1964) *Any graph G not containing an infinite clique has a simplicial decomposition $(G_\alpha)_{\alpha < \delta}$ into prime induced subgraphs G_α of G .*

Using Facts 1.2.2 and 1.2.4 it is now straightforward to prove the main result of this section:

Theorem 1.2.5 *Let G be an infinite graph.*

1. *If G has a tree decomposition into complete pieces, then G is chordal.*
2. *If G is chordal and contains no infinite clique, then G has a tree decomposition into complete pieces.*

Proof. 1: Let (T, B) be a tree decomposition of G into complete pieces. Let O be a cycle of length at least 4 in G . For each edge e of O choose a node $t_e \in T$ covering e . Let T' be the subtree of T spanned by the nodes $\{t_e \mid e \text{ an edge of } O\}$. Since T' is finite, we can apply Fact 1.2.2 to the subgraph of G covered by T' . Therefore O has a chord.

2: Let G be a chordal graph containing no infinite clique. By Fact 1.2.4, G has a simplicial decomposition $(G_\alpha)_{\alpha < \delta}$ into prime induced subgraphs. By Lemma 1.2.6 below, $(G_\alpha)_{\alpha < \delta}$ yields a tree decomposition (T, B) with pieces exactly the sets $V(G_\alpha)$. Since all the G_α are prime and chordal, they are complete by Lemma 1.2.7 below. $\square$

Lemma 1.2.6 *Let $(G_\alpha)_{\alpha < \delta}$ be a simplicial decomposition of G such that all G_α are finite. Then G has a tree decomposition (T, B) with pieces exactly the sets $V(G_\alpha)$.*

Proof. We prove this by induction on δ . Define (T^0, B^0) by setting $T^0 = (\{r\}, \emptyset)$ and $B_r^0 = \emptyset$. Then (T^0, B^0) is a tree decomposition of the empty graph $\bigcup_{\alpha < 0} G_\alpha$.

Let (T^β, B^β) be already defined, (T^β, B^β) a tree decomposition of $\bigcup_{\alpha < \beta} G_\alpha$. Since $K_\beta = G_\beta \cap \bigcup_{\alpha < \beta} G_\alpha$ is finite, by Proposition 1.1.8 there is a node $t \in V(T^\beta)$ such that $V(K_\beta) \subseteq B_t^\beta$. We now join $V(G_\beta)$ to the node t : Set

$$\begin{aligned} V(T^{\beta+1}) &:= V(T^\beta) \cup \{s\} \text{ for a new node } s \notin V(T^\beta), \\ E(T^{\beta+1}) &:= E(T^\beta) \cup \{\{s, t\}\} \text{ and} \\ B_t^{\beta+1} &:= B_t^\beta \text{ for all } t \in V(T^\beta), \quad B_s^{\beta+1} := V(G_\beta). \end{aligned}$$

It is easy to see that $(T^{\beta+1}, B^{\beta+1})$ is a tree decomposition of $\bigcup_{\alpha < \beta+1} G_\alpha$.

Let λ be a limit ordinal. For each $\alpha < \lambda$, let (T^α, B^α) be already defined, (T^α, B^α) a tree decomposition of $\bigcup_{\beta < \alpha} G_\beta$. Set $T^\lambda := \bigcup_{\alpha < \lambda} T^\alpha$ and $B^\lambda := \bigcup_{\alpha < \lambda} B^\alpha$. It is easy to see that (T^λ, B^λ) is a tree decomposition of $\bigcup_{\beta < \lambda} G_\beta$. Hence, (T^λ, B^λ) is a tree decomposition of G . $\square$

Lemma 1.2.7 *Let P be a prime chordal graph. Then P is complete.*

Proof. Suppose P is not complete. Then there are $p_1, p_2 \in V(P)$ with $\{p_1, p_2\} \notin E(P)$. Our aim is to find a clique in P that separate p_1 and p_2 , a contradiction to P being prime. The following two claims complete the proof:

Claim 1. There exists a minimal set $S \subseteq V(P)$ separating p_1 and p_2 .

Claim 2. A minimal set S separating p_1 and p_2 induces a clique in P .

Proof of Claim 1: The collection of subsets of $V(P) \setminus \{p_1, p_2\}$ which separate p_1 and p_2 is nonempty (as it contains $V(P) \setminus \{p_1, p_2\}$) and partially ordered by inclusion. Let $(S_i)_{i \in I}$ be a maximal chain, and let $S = \bigcap_{i \in I} S_i$. Then S separates p_1 and p_2 : If not, there is a path $p_1 = v_1, v_2, \dots, v_n = p_2$ in $P \setminus S$. Hence some S_{i_0} satisfies $S_{i_0} \cap \{v_i | i = 1, \dots, n\} = \emptyset$. But then S_{i_0} does not separate p_1 and p_2 , a contradiction to our assumption. Therefore S is a minimal set separating the vertices p_1 and p_2 .

Proof of Claim 2: If not, there are $s, t \in S$ such that $\{s, t\} \notin E(P)$. Let C_1 be the connected component of $P \setminus S$ containing p_1 and let C_2 be the connected component of $P \setminus S$ containing p_2 . Since S is minimal, both s and t have a neighbour in C_1 and a neighbour in C_2 . Let P_1 be the shortest path from s to t in $C_1 \cup \{s, t\}$ and let P_2 be the shortest path from s to t in $C_2 \cup \{s, t\}$. Then $P_1 \cup P_2$ is a cycle of length ≥ 4 without a chord, a contradiction. $\square$

Theorem 1.2.5, 2 does not hold for graphs G containing an infinite clique: [Ha64, p. 223] contains an example of a chordal graph that has no tree decomposition into complete pieces.

We will use Theorem 1.2.5 in Section 4.2, where we will generalise C. Thomassen's proof of the fact that an infinite graph has tree-width at most k if, and only if, all of its finite subgraphs have tree-width at most k .

As promised, we finish with an alternative characterisation of tree-width [Ha64].

Corollary 1.2.8 *Let G be a graph and $k > 0$ an integer. Then*

$$\begin{aligned} \text{tw}(G) \leq k \\ \iff \\ G \text{ has a triangulation } G' \text{ s. t. } G' \text{ contains no } (k+2)\text{-clique.} \end{aligned}$$

Proof. ' $\Rightarrow$ ': Let (T, B) be a tree decomposition of G of width $\text{width}(T, B) \leq k$. We define G' as follows.

Graph G'	
vertex set:	$V(G)$
edges:	$\{u, v\}$, where $\{u, v\} \subseteq B_t$ for a node $t \in T$

Then (T, B) is a tree decomposition of G' into complete pieces, and so G' is chordal by Theorem 1.2.5. Thus G' is a triangulation of G . Now for every complete subgraph $K \subseteq G'$, by Proposition 1.1.8 there is a node $t \in T$ such that $V(K) \subseteq B_t$. Hence $|V(K)| \leq |B_t| \leq k + 1$.

‘ $\Leftarrow$ ’: By Theorem 1.2.5, G' has a tree decomposition (T, B) into complete pieces, so $|B_t| \leq k + 1$. (T, B) is a tree decomposition of G of width at most k . $\square$

1.3 Tree-width and k -trees

k -trees were first defined in [AP81]. It is well known that for finite graphs, the subgraphs of k -trees are exactly the graphs of tree-width at most $k + 1$. In this section we prove this also for infinite graphs. We begin by refining the idea of Proposition 1.1.6 in order to show that every tree decomposition can be transformed into a particularly nice one. The aim of this section is to give another example where a well-known characterisation of tree decompositions can be maintained in the infinite case. The results of this section will not be used later.

1.3.1 Smooth tree decompositions

Definition 1.3.1 Let (T, B) be a tree decomposition of a graph G , of width at most k .

- (T, B) is *thick* if $|B_t| = k + 1$ for all $t \in T$.
- (T, B) is *small* if $B_s \not\subseteq B_t$ for any two distinct tree nodes $s, t \in T$.
- (T, B) is *smooth* if (T, B) is thick and $|B_s \cap B_t| = k$ holds for all tree edges $\{s, t\} \in E(T)$.

Obviously, every smooth tree decomposition is thick and small.

Lemma 1.3.2 Let $k < \omega$. If a graph G has a tree decomposition of width k , then G also has a thick tree decomposition of width k .

Proof. Let (T, B) be a width k tree decomposition of G . Then there exists a node $t_0 \in T$ such that $|B_{t_0}| = k + 1$. Set $(T, B^0) := (T, B)$.

Suppose we already know that (T, B^i) is a tree decomposition of G with $|B_t^i| = k + 1$ for all nodes $t \in T$ with distance $\leq i$ from t_0 . For any s with distance $i + 1$ from t_0 satisfying $|B_s| < k + 1$, add $(k + 1) - |B_s|$ elements from B_t to B_s , where t is the neighbour of s with distance i to t_0 . Thus we obtain a tree decomposition (T, B^{i+1}) of G , where $|B_{t_0}^{i+1}| = |B_s^{i+1}| = k + 1$ for all nodes $s \in T$ with distance $\leq i + 1$ from t_0 .

Now set $B^\omega := \bigcup_{i < \omega} B^i$. It is straightforward to check that (T, B^ω) is a thick tree decomposition of width k of G . $\square$

Lemma 1.3.3 If a graph G has a tree decomposition of width k , then G also has a thick and small tree decomposition of width k .

Proof. Let (T, B) be a tree decomposition of G of width k . By Lemma 1.3.2 we may assume that (T, B) is thick. Define (T', B') with $V(T') = V(T)/\sim$, where $\sim$ is the equivalence relation on $V(T)$ defined by

$$s \sim t : \Longleftrightarrow B_s = B_t.$$

We define $E(T')$ by setting

$$\{s', t'\} \in E(T') : \iff \text{there exist } s, t \in T \text{ such that } s \in s', t \in t' \text{ and } \{s, t\} \in E(T).$$

Finally, we set $B'_{t/\sim} := B_t$ for all $t \in T$. Since (T, B) is thick, for each $t_0 \in T$ there exists one and only one connected subset C of T such that all $t \in C$ satisfy $B_t = B_{t_0}$. It is easy to see that (T', B') is a small thick tree decomposition of G of width k . $\square$

Proposition 1.3.4 *If a graph G has a tree decomposition of width k , then G also has a smooth tree decomposition of width k .*

Proof. Let (T, B) be a width k tree decomposition of G . By Lemma 1.3.3 we may assume that (T, B) is small and thick. For each edge $\{s, t\} \in E(T)$ with $\ell := |B_s \cap B_t| < k$, we insert new nodes $s_1, \dots, s_{k-\ell}$ between s and t . Let $B_s = \{x_1, \dots, x_{k+1}\}$ and $B_t = \{x_1, \dots, x_\ell, y_{\ell+1}, \dots, y_{k+1}\}$. We set $B'_s := B_s$ and $B'_t := B_t$. Now define $B'_{s_i} := \{x_1, \dots, x_{k+1-i}, y_{k+2-i}, \dots, y_{k+1}\}$ for $i = 1, \dots, k - \ell$. Thus we obtain a smooth tree decomposition (T', B') of G of width k . $\square$

1.3.2 k -trees

Definition 1.3.5 *Let G be a graph, $v \notin V(G)$.*

For a subgraph $H \subseteq G$ we define a supergraph $G \cup \hat{H}^v \supseteq G$ (G with a cone over H) as follows.

Graph $G \cup \hat{H}^v$	
vertex set:	$V(G) \cup \{v\}$
edges:	e , where $e \in E(G)$ $\{v, u\}$, where $u \in V(H)$

$$G \cup \hat{H}^v := (V(G) \cup \{v\}, E(G) \cup \{\{v, u\} \mid u \in V(H)\}).$$

If $H = G$, we write $\hat{G}^v$ instead of $G \cup \hat{G}^v$.

Definition 1.3.6 *The class of k -trees is defined inductively as follows:*

(KT1) K_k , the complete graph on k vertices, is a k -tree.

(KT2) If G is a k -tree containing K_k as a subgraph and $v \notin V(G)$, then $G \cup \hat{K}_k^v$ is a k -tree.

(KT3) Let λ be a limit ordinal. If $(G_\alpha)_{\alpha < \lambda}$ is a family of k -trees s. t. $G_\alpha \subseteq G_\beta$ for $\alpha \leq \beta < \lambda$, then $\bigcup_{\alpha < \lambda} G_\alpha$ is a k -tree.

It easily follows from (KT3) that $\bigcup_{i \in J} G_i$ is a k -tree for every linearly ordered set J , if $G_i \subseteq G_j$ for $i < j$ and the graphs G_i are k -trees. It is also not hard to see that a k -tree that is a subgraph of another k -tree is in fact an induced subgraph.

Remark 1.3.7 *The 1-trees are exactly the trees.*

Proof. Let G be a 1-tree. We use induction to show that G is a tree:

(KT1): If $G = K_1$, then G is a tree. (KT2): If $G = G_0 \cup \hat{K}_1^v$ where $K_1 \subseteq G_0$ and G_0 is a tree, then G is a tree as well, since G is obtained from G_0 by connecting the new vertex v by a single edge to the only vertex of $K_1 \subseteq G_0$.

(KT3): If $G = \bigcup_{\alpha < \lambda} G_\alpha$, $G_\alpha \subseteq G_\beta$ for $\alpha \leq \beta < \lambda$ and each G_α is a tree, then G is a tree as well: Since all G_α are connected, G is connected. G contains no cycle, since a cycle in G would be contained in some G_α , $\alpha < \lambda$.

Conversely, let T be a tree. The set I of all induced subgraphs of T that are 1-trees is nonempty and ordered inductively by inclusion: The union of a linearly ordered sequence $(T_i)_{i \in J}$ in I is contained in T and is a 1-tree by definition. Now let M be a maximal element in I . Then $M = T$: Otherwise there is a node $t \in T \setminus V(M)$. Since T is connected, there is a path P leading from M to t . Let t_0 be the last node of P that is contained in $V(M)$ and let t_1 be the node after t_0 . We can join t_1 to M by the edge $\{t_0, t_1\}$, thus obtaining a 1-tree $M' \supsetneq M$ in T , a contradiction to the choice of M . $\square$

Lemma 1.3.8 *Let G be a k -tree and $v \notin V(G)$. Then $\hat{G}^v$ is a $(k+1)$ -tree.*

Proof. We prove this by induction:

(KT1): If $G = K_k$, then $\hat{G}^v = K_{k+1}$ is a $(k+1)$ -tree.

(KT2): If $G = G_0 \cup \hat{K}_k^u$ is a k -tree, $K_k \subseteq G_0$, then

$$\hat{G}^v = \widehat{G_0 \cup \hat{K}_k^u}^v = \hat{G}_0^v \cup \widehat{(\hat{K}_k^u)}^v = \hat{G}_0^v \cup \hat{K}_{k+1}^u$$

is a $(k+1)$ -tree, since $\hat{G}_0^v$ is a $(k+1)$ -tree by assumption.

(KT3): If $G = \bigcup_{\alpha < \lambda} G_\alpha$, $G_\alpha \subseteq G_\beta$ for $\alpha \leq \beta < \lambda$ and the $\hat{G}_\alpha^v$ are $(k+1)$ -trees for $\alpha \leq \beta < \lambda$, then $\hat{G}_\alpha^v \subseteq \hat{G}_\beta^v$ for $\alpha \leq \beta < \lambda$. Therefore $\hat{G}^v = \bigcup_{\alpha < \lambda} \hat{G}_\alpha^v$ is a $(k+1)$ -tree. $\square$

Corollary 1.3.9 *Let $\ell, k \geq 0$ be integers, $\ell \leq k$. Every ℓ -tree can be embedded in a k -tree.*

Proof. Use Lemma 1.3.8 $(k - \ell)$ times. $\square$

Theorem 1.3.10 *The graphs of tree-width $\leq k$ are exactly the (partial) subgraphs of k -trees.*

Proof of ‘ $\Rightarrow$ ’. Let $\ell := \text{tw}(G) \leq k$. By Corollary 1.3.9 it suffices to show that G is subgraph of an ℓ -tree. Let (T, B) be a smooth tree decomposition of width ℓ for G (such a tree decomposition exists by Proposition 1.3.4). Define a graph G' as follows.

Graph G'

vertex set: $V(G)$

edges: e , where $e \in E(G)$

$\{u, v\}$, where $\{u, v\} \subseteq B_t$ for a node $t \in T$

Obviously, (T, B) is also a tree decomposition of G' .

Claim: G' is an ℓ -tree.

Proof of the claim. T is a tree and hence, by Remark 1.3.7, T is a 1-tree. We prove the claim by induction on T .

(KT1): If $T = K_1 = (\{t\}, \emptyset)$, then G' is the complete graph on $\ell + 1$ vertices and therefore G' is an ℓ -tree.

(KT2): For $T_0 \subseteq T$ let G'_{T_0} denote the subgraph of G' induced by $\bigcup_{t \in V(T_0)} B_t$. Let $T = T_0 \cup \hat{K}_1^s$ with $s \notin V(T_0)$, and let G'_{T_0} be an ℓ -tree. Then

$$V(G') = V(G'_{T_0}) \cup B_s = V(G'_{T_0}) \cup \{v\}, \quad \text{for some } v \notin V(G'_{T_0}),$$

where the last equality holds because (T, B) is smooth. Furthermore

$$E(G') = E(G'_{T_0}) \cup \{\{u, v\} \mid u \in B_s, u \neq v\}.$$

$B_s \setminus \{v\}$ induces a complete graph K_l in G'_{T_0} . Note that an edge $\{u, v\}$ with $u \notin B_s$ cannot exist, since otherwise $\{u, v\}$ would be covered by some $t_0 \in V(T_0)$, a contradiction to $v \notin V(G'_{T_0})$. Hence $G' = G'_{T_0} \cup \hat{K}_l^v$ is an ℓ -tree.

(KT3): If $T = \bigcup_{\alpha < \lambda} T_\alpha$, $T_\alpha \subseteq T_\beta$ for $\alpha \leq \beta < \lambda$, and the G'_{T_α} are ℓ -trees, then $\bigcup_{\alpha < \lambda} G'_{T_\alpha}$ is an ℓ -tree by definition.

Altogether $G' = G'_T$ is an ℓ -tree. The proof of the other direction is given at the end of this section. $\square$

Definition 1.3.11 Let G be a k -tree. We define the construction tree T_G of G by:

$$\begin{aligned} V(T_G) &:= \{K \subseteq G \mid K \text{ is a } k\text{-clique or a } (k+1)\text{-clique}\}, \\ E(T_G) &:= \{\{K, K'\} \mid K \subset K' \subseteq G\}. \end{aligned}$$

Strictly speaking, the construction tree of G depends on G and k . This is because of the slightly pathological case of a $(k+1)$ -clique, which is a k -tree as well as a $(k+1)$ -tree. (We cannot avoid this by making the definition of k -trees more restrictive, since this would break Remark 1.3.7.)

Lemma 1.3.12 The construction tree T_G of a k -tree G is in fact a tree.

Proof. We prove this by induction on the k -tree G . Throughout the proof we use Remark 1.3.7 freely.

(KT1): If $G = K_k$, then $T_G = (\{K_k\}, \emptyset)$ is a tree.

(KT2): Let $G = G_0 \cup \hat{K}_k^v$ where $K_k \subseteq G_0$ and T_{G_0} is a tree. Then, by definition of the construction tree,

$$V(T_G) = V(T_{G_0}) \cup \{\hat{K}_k^v\} \cup \{K_k^1, \dots, K_k^k\}$$

where $K_k^i \neq K_k^j$ for $i \neq j$ and the $K_k^i \subseteq \hat{K}_k^v$ for $i = 1, \dots, k$ are the new k -cliques in $\hat{K}_k^v$, i. e. $v \in K_k^i$ for $i = 1, \dots, k$. Furthermore,

$$E(T_G) = E(T_{G_0}) \cup \{\{K_k, \hat{K}_k^v\}\} \cup \{\{\hat{K}_k^v, K_k^i\} \mid i = 1, \dots, k\}.$$

Therefore T_G is a tree.

(KT3): Let $G = \bigcup_{\alpha < \lambda} G_\alpha$, $G_\alpha \subseteq G_\beta$ for $\alpha \leq \beta < \lambda$, where the T_{G_α} are trees for $\alpha < \lambda$. Then $T_{G_\alpha} \subseteq T_{G_\beta}$ for $\alpha \leq \beta < \lambda$ and hence $T_G = \bigcup_{\alpha < \lambda} T_{G_\alpha}$ is a tree. $\square$

Lemma 1.3.13 Let G be a k -tree and T_G the construction tree of G . Then (T_G, B) with $B_K := V(K)$ for $K \in V(T_G)$ is a tree decomposition of G of width $\leq k$.

Proof. All $K \in V(T_G)$ satisfy $|B_K| = |V(K)| \leq k+1$ by definition of the construction tree. We now check the conditions for tree decompositions.

(TD1): Let $v \in V(G)$. It is easy to see that there is a k -clique $K_k \subseteq G$ with $v \in V(K_k)$. Therefore $v \in B_{K_k}$.

(TD2): Let $e \in E(G)$. It is easy to see that there is a k -clique $K_k \subseteq G$ with $e \subseteq V(K_k)$. Then $e \subseteq B_{K_k}$.

(TD3): Let $u \in V(G)$. By induction on G we show that $\{t \in V(T_G) \mid u \in B_t\}$ is connected:

- If $G = K_k$, then $V(T_G) = \{K_k\}$ and there is nothing to show.
- Let $G = G_0 \cup \hat{K}_k^v$, where $K_k \subseteq G_0$ and $(T_{G_0}, B \upharpoonright T_{G_0})$ satisfies (TD3). If $u \in B_{K_k^i}$ for one of the k new nodes $K_k^1, \dots, K_k^k \subseteq \hat{K}_k^v$, then $u \in B_{\hat{K}_k^v}$. Hence

it suffices to show: If $u \in B_K \cap B_{\hat{K}_k^v}$ for some $K \in V(T_{G_0})$, then $u \in B_s$ for all s on the path from K to $\hat{K}_k^v$.

Since $u \in B_K$, it follows that $u \neq v$. By the induction hypothesis, it suffices to show that $u \in B_{K_k}$ for the unique neighbour K_k of $\hat{K}_k^v$ in T_{G_0} , which is true since $u \neq v$.

- Let $G = \bigcup_{\alpha < \lambda} G_\alpha$, $G_\alpha \subseteq G_\beta$ for $\alpha \leq \beta < \lambda$. Since all (T_{G_α}, B_α) for $\alpha < \lambda$ satisfy (TD3), so does $(T_G, B) = \bigcup_{\alpha < \lambda} (T_{G_\alpha}, B_\alpha)$. $\square$

Proof of ‘ $\Leftarrow$ ’ in Theorem 1.3.10. Let G be a k -tree and $G_0 \subseteq G$ a subgraph. By Remark 1.1.4, it suffices to show that $\text{tw}(G) \leq k$. This is true by Lemma 1.3.13. $\square$

Chapter 2

f -hypertree-width of graphs

This chapter is motivated by hypergraphs, although in view of later application (Chapter 5) we work in a more general framework.

Let $H = (V, E)$ be a hypergraph, i. e. $E \subseteq \mathcal{P}_{<\omega}(V)$. A tree decomposition of H is a tree decomposition of the underlying graph $\underline{H}$ of H (cf. Definition 2.1.6). Alternatively, we can define a tree decomposition of H by translating the definition literally from graphs to hypergraphs, replacing the word ‘graph’ by ‘hypergraph’ and ‘edge’ by ‘hyperedge’. With Proposition 1.1.8 it is easy to see that the result is the same.

In the applications, however, it turns out that the cardinality $|B_t|$ of a piece B_t is not the best measure for its complexity: Any piece consisting of a single hyperedge should have small width, even though the cardinalities of such pieces are unbounded. This is achieved by the following ‘width function’:

$$\begin{aligned} c_H^{\text{mon}} : \mathcal{P}_{<\omega}(V(H)) &\rightarrow \mathbb{R} \cup \{\infty\}, \\ X &\mapsto \text{the least integer } n \text{ such that } X \text{ is contained in the} \\ &\quad \text{the union of } n \text{ hyperedges of } H. \end{aligned}$$

Given a width function f such as c_H^{mon} , we can proceed to define the f -width of a tree decomposition (T, B) as $f\text{-width}(T, B) = \sup\{f(B_t) \mid t \in T\}$. The *exact f -hypertree-width* of a graph (or hypergraph) G is

$$f\text{-ehw}(G) = \inf\{f\text{-width}(T, B) \mid (T, B) \text{ a tree decomposition of } G\}.$$

In the case of $f = c_H^{\text{mon}}$ it is easy to see that we get $c_H^{\text{mon}}\text{-ehw}(H) = \text{ghw}(H)$, the *generalised hypertree-width* of H as defined by G. Gottlob, N. Leone and F. Scarcello in [GLS01b].

For technical and historical reasons (which again have to do with the applications) we work with two other, more complicated notions rather than with exact f -hypertree-width. These are f -hypertree-width, f -hw, and generalised f -hypertree-width, f -ghw (cf. Definition 2.1.3). If f , like c_H^{mon} , is a monotone width function, i.e.

$$X \subseteq Y \subseteq V(G) \implies f(X) \leq f(Y),$$

then all three definitions coincide:

$$f\text{-ghw}(G) = f\text{-hw}(G) = f\text{-ehw}(G).$$

For non-monotone width functions, we only have

$$f\text{-ghw}(G) \leq f\text{-hw}(G) \leq f\text{-ehw}(G).$$

In Section 2.2 we characterise $f\text{-hw}(G)$ as the minimal number k such that in a monotone robber and cops game in the style of P. D. Seymour and R. Thomas [ST93], the cops can win if they are allowed to occupy sets X such that $f(X) \leq k$.

In Section 2.3, we slightly generalise some standard bounded tree-width and bounded hypertree-width methods in order to get tractable cases of the homomorphism problem. Here it should become clear why we are interested in both $f\text{-hw}$ and $f\text{-ghw}$.

Since we can regard every hypergraph H as its underlying graph $\underline{H}$ equipped with the width function c_H^{mon} it makes sense to examine more generally graphs equipped with arbitrary width functions, and also classes of ‘equipped’ graphs. Hypertree-width and generalised hypertree-width can be regarded as invariants of equipped graphs. In Section 2.4 we define some more invariants for equipped graphs that resemble graph invariants such as bramble-number, linkedness, branch-width, and we prove that for certain classes of equipped graphs they are all equivalent in a strong sense.

2.1 Definitions and some observations

2.1.1 Hypertree decompositions and width functions

Definition 2.1.1 *Let G be a graph.*

- A generalised hypertree decomposition of G is a triple (T, B, C) , such that
 - (HD1) (T, B) is a tree decomposition of G ,
 - and $C = (C_t)_{t \in T}$ is a family of finite subsets of $V(G)$, the guards¹ of (T, B, C) , such that
 - (HD2) Every tree node $t \in T$ satisfies $B_t \subseteq C_t$.
- (T, B, C) is a hypertree decomposition of G if the following condition is also satisfied:
 - (HD3) Every tree node $t \in T$ satisfies $C_t \cap B_{T_t} \subseteq B_t$. (Recall that $B_{T_t} = \bigcup_{s \in V(T_t)} B_s$.)

Thus the axioms for hypertree decompositions are (TD1), (TD2), (TD3), (HD2), (HD3). Recall from Definition 1.1.2 that the pieces B_t must be finite. Note that for every tree decomposition (T, B) we get a hypertree decomposition (T, B, B) , and that every hypertree decomposition is a generalised hypertree decomposition.²

Definition 2.1.2 *A width function on a graph G is a function*

$$f : \mathcal{P}_{<\omega}(V(G)) \rightarrow \mathbb{R} \cup \{\infty\}.$$

Definition 2.1.3 *Let G be a graph, let f be a width function on G , and let (T, B, C) be a hypertree decomposition of G .*

- The f -width of (T, B, C) is

$$f\text{-width}(T, B, C) = \sup\{f(C_t) \mid t \in T\}.$$

¹Readers already acquainted with hypertree decompositions of hypergraphs as defined in [GLS02] will note that our guards C_t are sets of graph *vertices*, not of edges. If C'_t is a guard in [GLS02], then $C_t = \bigcup C'_t$.

²The reader may wonder what a ‘hypertree’ is. Should these decompositions not be called ‘hyper-treedecompositions’? The rationale behind this terminology is that a ‘tree’ in the sense of tree decompositions is a pair (T, B) and a ‘hypertree’ is a triple (T, B, C) .

- The f -hypertree-width of G is

$$f\text{-hw}(G) = \inf \{ f\text{-width}(T, B, C) \mid (T, B, C) \text{ a hypertree decomposition of } G \}.$$

- The generalised f -hypertree-width of G is

$$f\text{-ghw}(G) = \inf \{ f\text{-width}(T, B, C) \mid (T, B, C) \text{ a generalised hypertree decomposition of } G \}.$$

The ‘generalised’ f -hypertree-width of a graph is, of course, not more general but just smaller than its f -hypertree-width: $f\text{-ghw}(G) \leq f\text{-hw}(G)$. Also note that if G is finite or if f actually takes values in a well-ordered subset of $\mathbb{R} \cup \{\infty\}$, such as $\omega \cup \{\infty\}$, then the infima in the definitions of $f\text{-hw}(G)$ and $f\text{-ghw}(G)$ are attained.

The notion of exact hypertree-width defined in the introduction of this chapter has an analogous characterisation in terms of ‘exact’ hypertree decompositions, i.e. hypertree decompositions of the form (T, B, B) .

Example 2.1.4 Let G be a graph. The cardinality function

$$\begin{aligned} \text{card} : \mathcal{P}_{<\omega}(V(G)) &\rightarrow \mathbb{R} \cup \{\infty\}, \\ X &\mapsto |X|, \end{aligned}$$

is a width function on G , and

$$\text{card-hw}(G) = \text{card-ghw}(G) = \text{tw}(G) + 1.$$

Once again, to a reader who is mainly interested in finite graphs (and finite hypergraphs), it may seem somewhat unnatural to admit infinite hypertree decompositions for finite graphs. But—as in the case of tree decompositions—it is harmless:

Theorem 2.1.5 Let G be a finite graph, let f be a width function for G and let $k \in \mathbb{R}$.

1. If $f\text{-ghw}(G) = k$, then there is a finite generalised hypertree decomposition (T, B, C) with $f\text{-width}(T, B, C) = k$.
2. If $f\text{-hw}(G) = k$, then there is a finite hypertree decomposition (T, B, C) with $f\text{-width}(T, B, C) = k$.

Proof. First note that the f -width of a (generalised) hypertree decomposition, although defined as a supremum, is in fact a maximum in case of finite G , and that $f\text{-ghw}(G)$ and $f\text{-hw}(G)$ are in fact minima. Therefore we have a (generalised) hypertree decomposition (T, B, C) of G such that $f\text{-width}(G) = k$. Transform (T, B) into a tree decomposition (T', B') as in Proposition 1.1.6. Then T' is finite, and for every node $t' \in T'$ there exists a node $t \in T$ with $B'_{t'} = B_t$. Choose such a node t closest to the root and define $C'_{t'} := C_t$. Then (T', B', C') is a generalised hypertree decomposition of G . Moreover, if (T, B, C) satisfies (HD3), then so does (T', B', C') . $\square$

2.1.2 Application to hypergraphs

Our only example of f -hw and f -ghw so far is tree-width (Example 2.1.4). To give a second type of examples, we give the definitions of a *hypergraph* and its *cover number*. These will be investigated in detail in Chapter 3.

Definition 2.1.6

- A hypergraph is a pair $H = (V(H), E(H))$, consisting of
 - a nonempty set $V(H)$ of vertices, and
 - a set $E(H) \subseteq \mathcal{P}_{<\omega}(V(H))$ of finite subsets of $V(H)$, the hyperedges of H (which we will usually just call ‘edges’).
- H_0 is a subhypergraph of H , if $V(H_0) \subseteq V(H)$ and $E(H_0) \subseteq E(H)$.
- Let $X \subseteq V(H)$. Then the subhypergraph of H induced by X is the following hypergraph $H[X]$:

Hypergraph $H[X]$	
vertex set:	X
edges:	$e \cap X$, where $e \in E(H)$

- H_0 is an induced subhypergraph of H if $H_0 = H[X]$ for some $X \subseteq V(H)$.
- For a hypergraph H the underlying graph of H is the following graph $\underline{H}$:

Hypergraph $\underline{H}$	
vertex set:	$V(H)$
edges:	$\{v, w\}$, where $v \neq w$ and there exists an edge $h \in E(H)$ such that $\{v, w\} \subseteq h$

$\underline{H}$ is also called the primal graph or the Gaifman graph of H .

- A hypergraph H is connected if $\underline{H}$ is connected.
- A hypergraph H is tame if H has no isolated vertices, i. e. every vertex $v \in V(H)$ is contained in a hyperedge $h \in E(H)$.

We say that f is a width function on the hypergraph H , if f is a width function on $\underline{H}$.

Example 2.1.7 Here is the hypergraph shown in Figure 2.1:

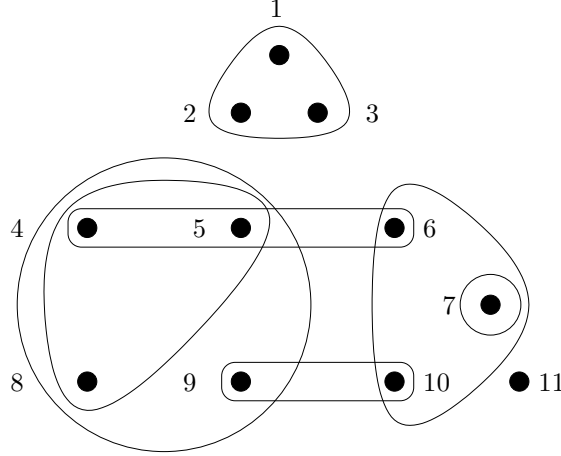
Hypergraph H	
vertices:	$1, 2, \dots, 11$
edges:	$\{1, 2, 3\}, \{4, 5, 6\}, \{4, 5, 8\}, \{4, 5, 8, 9\}$ $\{6, 7, 10\}, \{7\}, \{9, 10\}$

Definition 2.1.8 We consider the following width function on hypergraphs H :

$$c_H : \mathcal{P}_{<\omega}(V(H)) \rightarrow \mathbb{R} \cup \{\infty\},$$

$$X \mapsto \inf \{ |Y| \mid Y \subseteq E(H), X = \bigcup Y \}.$$

We call $c_H(X)$ the cover number of X .

Figure 2.1: The hypergraph H from Example 2.1.7.**Definition 2.1.9**

- A tree decomposition of a hypergraph H is a tree decomposition of $\underline{H}$.
- The tree-width of H is the tree-width of $\underline{H}$.
- A (generalised) hypertree decomposition of H is a (generalised, resp.) hypertree decomposition of $\underline{H}$.
- For a width function f of H , we define $f\text{-hw}(H) := f\text{-hw}(\underline{H})$ and $f\text{-ghw}(H) := f\text{-ghw}(\underline{H})$.

Alternatively, we could define the notions of tree decomposition, hypertree decomposition, tree-width of H and f -hypertree-width of H by translating the definitions literally from graphs to hypergraphs, replacing the word ‘graph’ by ‘hypergraph’ and ‘edge’ by ‘hyperedge’. With Proposition 1.1.8 it is easy to see that the result would be the same. For some of the other f -invariants defined below in Section 2.4 this will not be true.

Example 2.1.10 Let H be a finite hypergraph. Then

- $c_H\text{-hw}(H)$ equals the hypertree-width of H as defined in [GLS02].
- $c_H\text{-ghw}(H)$ equals the generalised hypertree-width of H as defined in [GLS02].

Remark 2.1.11 Let H' be an induced subhypergraph of H . Then

- $c_{H'}\text{-hw}(H') \leq c_H\text{-hw}(H)$, and
- $c_{H'}\text{-ghw}(H') \leq c_H\text{-ghw}(H)$.

Proof. Let (T, B, C) be a (generalised) hypertree decomposition of H satisfying $c_H\text{-width}(T, B, C) \leq k$. Set $B'_t := B_t \cap V(H')$ and $C'_t := C_t \cap V(H')$. It is easy to see that (T, B', C') is a (generalised) hypertree decomposition of H' . Moreover, since H' is induced, we have $c_H\text{-width}(T, B', C') \leq k$, because $c_{H'}(C'_t) \leq c_H(C_t)$ for all tree nodes $t \in T$. $\square$

This is not true for arbitrary subhypergraphs (in contrast to Remark 1.1.4):

Remark 2.1.12 For an integer $n > 0$ consider the following hypergraph H :

Hypergraph H	
vertices:	$1, \dots, 2n$
edges:	$\{1, \dots, 2n\}$ $\{i, j\}, \text{ where } i < j, i, j \in \{1, \dots, 2n\}$

The hypergraph H has the following subhypergraph H' :

Hypergraph H'	
vertices:	$1, \dots, 2n$
edges:	$\{i, j\}, \text{ where } i < j, i, j \in \{1, \dots, 2n\}$

For H and H' we have

- $c_H\text{-ghw}(H) = c_H\text{-hw}(H) = 1$, and
- $c_H\text{-ghw}(H') = c_H\text{-hw}(H') = n$.

Proof. The first statement is obvious. For the second statement, note that $\underline{H'}$ is a clique. Thus, by Proposition 1.1.8, every tree decomposition (T, B) of H' has a piece B_t with $V(H') \subseteq B_t$. The rest is left to the reader. $\square$

2.1.3 Monotone width functions

If f is a width function on G and $G' \subseteq G$ a subgraph of G , then the restriction of f to $V(G')$ is a width function on G' ; and one might expect that $f\text{-hw}(G') \leq f\text{-hw}(G)$. However, this is not the case in general: consider the width function $f(X) = -|X|$ on an arbitrary graph G . This is the reason for the following slightly complicated definition.

Definition 2.1.13 Let f be a width function on the graph G and let G' be a subgraph of G . The function f gives rise to a width function $f_{G'}$ on G' as follows.

$$f_{G'} : \mathcal{P}_{<\omega}(V(G)) \rightarrow \mathbb{R} \cup \{\infty\},$$

$$X \mapsto \inf \{f(Y) \mid Y \subseteq V(G), |Y| < \omega, \text{ and } X = V(G') \cap Y\}.$$

If H is a hypergraph and $H' \subseteq H$ is an induced subhypergraph, then $c_{H'} = (c_H)_{H'}$. Hence the following generalises Remark 2.1.11.

Remark 2.1.14 Let f be a width function on the graph G , let G' be a subgraph of G . Then

- $f_{G'}\text{-hw}(G') \leq f\text{-hw}(G)$, and
- $f_{G'}\text{-ghw}(G') \leq f\text{-ghw}(G)$.

Proof. We can restrict a (generalised) hypertree decomposition (T, B, C) by setting $B'_t = B_t \cap V(G')$ and $C'_t = C_t \cap V(G')$. Clearly, $f_{G'}(C'_t) \leq f(C_t)$. $\square$

Definition 2.1.15 A width function f on the graph G is monotone, if for finite subsets $X, Y \subseteq V(G)$ we have

$$X \subseteq Y \implies f(X) \leq f(Y).$$

Example 2.1.16

- Let G be a graph. The cardinality function card is monotone.

- In general, the cover number of a hypergraph is not monotone. For example, the hypergraph $H = (\{1, 2\}, \{\{1, 2\}\})$ satisfies

$$\infty = c_H(\{1\}) > c_H(\{1, 2\}) = 1.$$

- If H is a hypergraph, then c_H is monotone if, and only if, H is simplicial, i. e. every subset of a hyperedge is itself a hyperedge.

For monotone width functions f the above complicated definition of $f_{G'}$ is not necessary:

Remark 2.1.17 Let f be a monotone width function on the graph G , and let G' be a subgraph of G . Then $f_{G'} = f \upharpoonright \mathcal{P}_{<\omega}(V(G'))$.

Proof. Let $X \subseteq V(G)$ be a finite subset. Then

$$\begin{aligned} f_{G'}(X) &= \inf \{f(Y) \mid Y \subseteq V(G), |Y| < \omega, \text{ and } X = V(G') \cap Y\} \\ &= f(X), \end{aligned}$$

where the last equality holds because f is monotone. $\square$

For graphs with monotone width functions, the hypertree decompositions can be chosen to have a very simple form: If (T, B, C) is a (generalised) hypertree decomposition for G of f -width at most k , then (T, B, B) is also a (generalised) hypertree decomposition of f -width at most k .

Remark 2.1.18 Let f be a monotone width function on the graph G . Then

$$f\text{-ghw}(G) = f\text{-hw}(G).$$

Proof. Since every hypertree decomposition is a generalised hypertree decomposition, we have $f\text{-ghw}(G) \leq f\text{-hw}(G)$. Conversely, let $f\text{-ghw}(G) \leq k$. By monotonicity of f we may assume that this is witnessed by a generalised hypertree decomposition (T, B, B) . But (T, B, B) satisfies condition (HD3) of Definition 2.1.1 and thus is a hypertree decomposition. $\square$

Below in Section 3.3.1 we will see examples where

$$f\text{-ghw}(G) < f\text{-hw}(G).$$

Every width function can be ‘made monotone’ in the following way.

Definition 2.1.19 Given a width function f on the graph G , we define the width function f^{mon} on G as follows.

$$\begin{aligned} f^{\text{mon}} : \mathcal{P}_{<\omega}V(G) &\rightarrow \mathbb{R} \cup \{\infty\}, \\ X &\mapsto f_{G[X]}(X) = \inf \{f(Y) \mid X \subseteq Y \subseteq V(G), |Y| < \omega\}. \end{aligned}$$

Obviously, if f only takes values in $\omega \cup \{\infty\}$, then the infimum in the definition above is actually a minimum. For example this is the case if $f = c_H$.

Lemma 2.1.20 Let f be a width function on G . Then

1. $f^{\text{mon}} \leq f$,
2. f^{mon} is monotone,
3. $f^{\text{mon}} = \sup \{g \mid g \text{ monotone width function on } G \text{ and } g \leq f\}$,

4. If f is monotone, then $f = f^{\text{mon}}$.

Proof. 1: For $X \subseteq V(G)$, we have

$$f^{\text{mon}}(X) = \inf \{f(Y) \mid X \subseteq Y \subseteq V(G)\} \leq f(X).$$

2: Let $X \subseteq Y \subseteq V(G)$. Then

$$\{f(Z) \mid X \subseteq Z \subseteq V(G)\} \supseteq \{f(Z) \mid Y \subseteq Z \subseteq V(G)\},$$

and thus

$$\begin{aligned} f^{\text{mon}}(X) &= \inf \{f(Z) \mid X \subseteq Z \subseteq V(G)\} \\ &\leq \inf \{f(Z) \mid Y \subseteq Z \subseteq V(G)\} = f^{\text{mon}}(Y). \end{aligned}$$

3: Let g be a monotone width function on G such that $g \leq f$. For $X \subseteq Y \subseteq V(G)$ we have $g(X) \leq f(Y)$. Thus $g(X) \leq \inf \{f(Y) \mid X \subseteq Y \subseteq V(G)\} = f^{\text{mon}}(X)$, and the statement follows from 2. 4: Follows from 3. $\square$

Proposition 2.1.21 *Let G be a graph and let f be a width function on G . Then*

$$f^{\text{mon}}\text{-hw}(G) = f^{\text{mon}}\text{-ghw}(G) = f\text{-ghw}(G).$$

Proof. By Lemma 2.1.20, f^{mon} is monotone, and the first equality follows from Remark 2.1.18. By Part 1 of Lemma 2.1.20, $f^{\text{mon}} \leq f$. Thus $f^{\text{mon}}\text{-ghw}(G) \leq f\text{-ghw}(G)$. Conversely, let $f^{\text{mon}}\text{-ghw}(G) \leq k$. We have to show that for all $\varepsilon > 0$ the inequality $f\text{-ghw}(G) \leq k + \varepsilon$ holds. Let (T, B, C) be a generalised hypertree decomposition witnessing that $f^{\text{mon}}\text{-ghw}(G) \leq k + \frac{\varepsilon}{2}$. Since each $t \in T$ satisfies $f^{\text{mon}}(C_t) \leq k + \frac{\varepsilon}{2}$, by definition of f^{mon} there exists a set C'_t with $C_t \subseteq C'_t \subseteq V(G)$ and $f(C'_t) \leq (k + \frac{\varepsilon}{2}) + \frac{\varepsilon}{2}$. Thus (T, B, C') witnesses that $f\text{-ghw}(G) \leq k + \varepsilon$. $\square$

For completeness we also state the following easy remarks.

Remark 2.1.22 *Let f be a width function on the graph G and let G' be a subgraph of G . Then*

$$(f^{\text{mon}})_{G'} = (f_{G'})^{\text{mon}}.$$

Proof. Let X be a finite subset of $V(G')$. Then

$$\begin{aligned} (f^{\text{mon}})_{G'}(X) &= \inf_{\substack{Y \subseteq V(G) \\ Y \cap V(G') = X}} \inf_{\substack{Z \subseteq V(G) \\ Z \supseteq Y}} f(Z) = \inf_{\substack{Z \subseteq V(G) \\ Z \supseteq X}} f(Z) \\ &= \inf_{\substack{Y \subseteq V(G') \\ Y \supseteq X}} \inf_{\substack{Z \subseteq V(G) \\ Z \cap V(G') = Y}} f(Z) = (f_{G'})^{\text{mon}}(X). \end{aligned}$$

$\square$

Remark 2.1.23 *Let f be a width function on the graph G . Let $G = \bigcup_{\beta < \alpha} G[C_\beta]$, where the C_β , for $\beta < \alpha$, are the connected components of the graph G (for some ordinal α). Then*

$$f\text{-hw}(G) = \sup \{f_{G[C_\beta]}\text{-hw}(G[C_\beta]) \mid \beta < \alpha\}.$$

2.2 A game for f -hypertree-width

In [ST93], P. D. Seymour and R. Thomas defined a game in which a number of cops must catch a robber on a graph. They also defined a monotone variant of the game, in which the cops have to make sure that the robber's 'escape space' always decreases. They proved that on a given graph G the minimum number of cops necessary to catch the robber in the robber and cops game equals the number of cops necessary to catch the robber in the monotone game variant. Furthermore they showed that these invariants are equal to $\text{tw}(G) + 1$.

In this section we define very similar games. Where P. D. Seymour and R. Thomas bounded $|X|$ for a cop position $X \subseteq V(G)$, we bound $f(X)$ for an arbitrary width function f on G . In this respect the generalisation is very straightforward. However, in order to make this game behave as expected on infinite graphs, we need to be a bit more careful.

For arbitrary width functions our robber and cops game and its monotone variant may not define the same f -invariant, even on finite graphs. Examples of this phenomenon will be given in Sections 3.2 and 3.3.

But we will see in Section 2.2.3s that the number of cops necessary to catch the robber monotonely on a graph G equals $f\text{-hw}(G)$.

2.2.1 Robber and cops

Let G be a graph, let f be a width function on G and $k \in \mathbb{R}$. Informally, the robber and cops game on G (with game parameters f and k) is played by two players, the cop player and the robber player, on the graph G . The cop player controls arbitrarily many cops and the robber player controls the robber. Both the cops and the robber move on the vertices of G . The cops choose finite subsets $X \subseteq V(G)$ with $f(X) \leq k$. In each move, some of the cops fly in helicopters to new vertices. The robber sees where the cops will be landing and quickly tries to escape by running arbitrarily fast along paths of G , not being allowed to run through a cop. If G is finite, the cop player's objective is to land a cop via helicopter on the vertex occupied by the robber. The robber player tries to elude capture.

A delicate question is how to determine the winner after an infinite chase. Is it possible to capture the robber through an infinite sequence of moves? If we were only interested in finite graphs we could simply say that the winner of an infinite play is always the robber. Instead we define: The cops win an infinite play if for every vertex $v \in V(G)$ there is a move after which the robber can never reach v again.

In the rest of this section we will make this informal description precise.

Definition 2.2.1 *Let G be a graph, let f be a width function on G and $k \in \mathbb{R}$. The robber and cops game $\text{RC}(G, f, k)$ on G and its 'monotone' variant, the monotone robber and cops game $\text{RC}_{\text{mon}}(G, f, k)$, are the games described below.*

A *play* of $\text{RC}(G, f, k)$ is a (finite or countably infinite) sequence

$$(X_0, r_0, X_1, r_1, X_2, r_2, \dots)$$

subject to the following conditions:

- (C1) Each *cop move* X_i is a finite subset $X_i \subseteq V(G)$ satisfying $f(X_i) \leq k$.
- (R1) Each *robber move* r_i is a graph vertex $r_i \in V(G) \setminus X_i$.
- (R2) For all indices $i > 0$ the vertices r_{i-1} and r_i are connected by a path in $G \setminus (X_{i-1} \cup X_i)$.

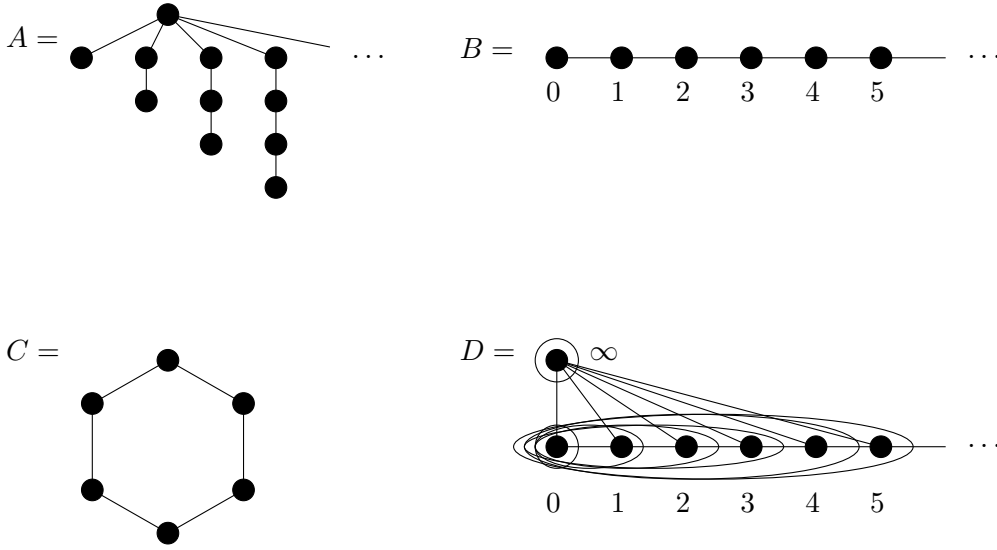


Figure 2.2: $\text{RC}(G, f, 1)$. For A, B, C : $f(X) := 1$ if X is a graph edge, and $f(X) := \infty$ otherwise. For D : $f(X) := 1$ if X is one of the ellipses, and $f(X) := \infty$ otherwise. The cops can win on A and B , but not on C and D .

The game starts with the empty play, and the *cop player* (also called *the cops*) moves first. A move of the cop player consists in adding a cop move X_n to the play

$$\pi = (X_0, r_0, X_1, r_1, \dots, X_{n-1}, r_{n-1})$$

in such a way that the resulting sequence

$$\pi \frown X_n = (X_0, r_0, X_1, r_1, \dots, X_{n-1}, r_{n-1}, X_n)$$

is again a play. Similarly, a move of the *robber player* consists in adding a robber move r_n to the play

$$\pi = (X_0, r_0, X_1, r_1, \dots, X_n)$$

in such a way that the resulting sequence

$$\pi \frown r_n = (X_0, r_0, X_1, r_1, \dots, X_n, r_n)$$

is a play. The game is played infinitely long, or until a player cannot move any more. Thus after finishing an instance of the game we have a *maximal play*: A play that is not a proper initial subsequence of any other play.

The winner of a finite maximal play is simply the last player who moved. If we considered only finite graphs, we could say that every infinite play is won by the robber. But applying this definition to infinite graphs, the cop player could only win case A in Figure 2.2. However, since we want to characterise f -hw with this game, the cop player should be able to win case B as well, by the following series of moves: 01, 12, 23, 34, $\dots$. Of course we need to make sure that the cop player cannot win cases C or D . This is achieved by the following *winning condition* for $\text{RC}(G, f, k)$:

Let $(X_0, r_0, X_1, r_1, X_2, r_2, \dots)$ be an infinite play. Suppose for every graph vertex $v \in V(G)$ there exists an index $n < \omega$ such that for all $i > n$ the vertices v and r_i are not connected in $G \setminus X_i$. Then the cop player wins. Otherwise the robber player wins.

It is often convenient to allow players to give up. In this case the last player who moved wins. Thus we can extend the above definition to arbitrary (possibly

non-maximal) plays by saying that the cop player wins every finite play of odd length (not just the maximal ones), and the robber player wins every finite play of even length.

The *monotone robber and cops game* $\text{RC}_{\text{mon}}(G, f, k)$ is the variant of the game in which plays are further restricted by the following monotonicity condition on the cop moves:

(C2) Let C be the connected component of r_i in $G \setminus X_i$. Then C is also a connected component in $G \setminus (X_i \cap X_{i+1})$.

Everything else is defined as in $\text{RC}(G, f, k)$, so the plays for $\text{RC}_{\text{mon}}(G, f, k)$ form a subset of the plays for $\text{RC}_{\text{mon}}(G, f, k)$.

Definition 2.2.2 For a play $\pi = (X_0, r_0, \dots, X_n, r_n)$ on G of even length we define the robber's escape space

$$R_\pi = \{v \in V(G) \mid v \text{ and } r_n \text{ are connected in } G \setminus X_n\}.$$

For $\pi = ()$, the unique play of length 0, we define $R_\pi = V(G)$.

Instead of condition (C2) we can consider the following condition on plays $\pi = (X_0, r_0, X_1, r_1, \dots)$:

(C2') $R_{\pi_0} \supseteq R_{\pi_1} \supseteq R_{\pi_2} \supseteq \dots$, where $\pi_i = (X_0, r_0, X_1, r_1, \dots, X_i, r_i)$.

The two conditions are equivalent in the following sense:

Remark 2.2.3

1. If a play π satisfies (C2), then it also satisfies (C2').
2. If a play π does not satisfy (C2), then there is an $i < \omega$ and an $r'_i \in V(G)$ such that $\pi' = (X_0, r_0, X_1, r_1, \dots, X_i, r'_i)$ is a play not satisfying (C2').

Proof: 1: Obvious. 2: If π does not satisfy (C2), then there is an $i < \omega$ such that R_{π_i} is not a connected component of $G \setminus (X_i \cap X_{i+1})$. Let C be the component of $G \setminus (X_i \cap X_{i+1})$ containing R_{π_i} . Then there exists a vertex $r'_i \in C \setminus R_{\pi_i}$, the robber can reach r'_i during the flight, and the play $\pi' = (X_0, r_0, X_1, r_1, \dots, X_i, r'_i)$ does not satisfy (C2). $\square$

Some definitions of the monotone robber and cops game require

$$R_{\pi_0} \supsetneq R_{\pi_1} \supsetneq R_{\pi_2} \supsetneq \dots$$

It is easy to see that this further 'strictly monotone' variant of the game is equivalent to $\text{RC}_{\text{mon}}(G, f, k)$ in the sense that if the cops can win $\text{RC}_{\text{mon}}(G, f, k)$, then they can also win the strictly monotone variant (while the converse is obvious).

2.2.2 Strategies

In this section we fix a game $\text{RC}(G, f, k)$ or $\text{RC}_{\text{mon}}(G, f, k)$. Since we are dealing with infinite plays we will give a precise definition of what it means for the cop player to have a winning strategy.

Definition 2.2.4 A strategy for the cop player is a function σ associating to every finite play π of even length a finite subset $X = \sigma(\pi) \subseteq V(G)$. A play $(X_0, r_0, X_1, r_1, \dots)$ is consistent with σ , if for every proper initial sequence

$$\pi = (X_0, r_0, X_1, r_1, \dots, X_n, r_n)$$

of even length we have $X_{n+1} = \sigma(\pi)$.

Similarly, a strategy for the robber player is a function ρ associating to every finite play π of odd length a vertex $r \in V(G)$. A play $(X_0, r_0, X_1, r_1, \dots)$ is consistent with ρ , if for every proper initial sequence

$$\pi = (X_0, r_0, X_1, r_1, \dots, X_n)$$

of odd length we have $r_n = \rho(\pi)$.

Remark 2.2.5 Given a cop strategy σ and a robber strategy ρ there is a unique play π that is maximal subject to the condition that it is consistent with σ and ρ . $\square$

The maximal play consistent with σ and ρ need not be a maximal play. This is because one of the strategies may prescribe an illegal move. Note that our winning condition does the right thing in this case: the last player to move wins, and choosing an illegal move is the same thing as giving up.

Definition 2.2.6 We denote the maximal play consistent with σ and ρ by $\pi(\sigma, \rho)$. The cop strategy σ is a winning strategy (for the cop player) if, for every robber strategy ρ , the cop player wins the play $\pi(\sigma, \rho)$. Similarly, the robber strategy ρ is a winning strategy (for the robber player) if the robber player wins $\pi(\sigma, \rho)$ for every cop strategy σ .

Obviously, at most one player has a winning strategy. It is open, whether one of the players always has a winning strategy. For finite graphs it is easy to see that this is in fact the case.

From Remark 2.2.3 it follows that if we define $\text{RC}'_{\text{mon}}(G, f, k)$ with (C2') instead of (C2), then the cops have a winning strategy for $\text{RC}'_{\text{mon}}(G, f, k)$ if, and only if, they have a winning strategy for $\text{RC}_{\text{mon}}(G, f, k)$. We will often use this fact tacitly.

Definition 2.2.7 A strategy σ for the cop player is positional, if it satisfies the following condition:

$$\sigma((X_0, r_0, \dots, X_n, r_n)) = \sigma((X'_0, r'_0, \dots, X'_n, r'_n))$$

whenever $X_n = X'_n$ and $r_n = r'_n$.

A strategy σ for the robber player is positional, if it satisfies the following condition:

$$\rho((X_0, r_0, \dots, X_n, r_n, X_{n+1})) = \rho((X'_0, r'_0, \dots, X'_n, r'_n, X'_{n+1}))$$

whenever $X_n = X'_n$, $r_n = r'_n$ and $X_{n+1} = X'_{n+1}$.

It is not hard to check that for finite graphs G , if the cops have a winning strategy for $\text{RC}(G, f, k)$, then they have a positional winning strategy for $\text{RC}(G, f, k)$. We omit this because we will not use this fact.

While we do not know if it is true for infinite graphs, we will show in Section 2.2.3 that the corresponding statement for $\text{RC}_{\text{mon}}(G, f, k)$ holds for arbitrary graphs.³

Definition 2.2.8 Let G be a graph and let f be a width function on G .

- The f -cop-width of G is

$$f\text{-cw}(G) := \inf \{k \mid \text{the cops have a winning strategy in } \text{RC}(G, f, k)\} \in \mathbb{R} \cup \{\infty\}.$$

³The definition of positionality for robber strategies was only included for completeness, and we will not use it.

- The monotone f -cop-width of G is

$$f\text{-cw}_{\text{mon}}(G) := \inf \{k \mid \text{the cops have a winning strategy in } \text{RC}_{\text{mon}}(G, f, k)\} \in \mathbb{R} \cup \{\infty\}.$$

Example 2.2.9 Let G be a finite graph and $k > 0$ an integer. Then the game $\text{RC}(G, \text{card}, k)$ is the robber and k cops game as defined in [ST93], and the game $\text{RC}_{\text{mon}}(G, \text{card}, k)$ is its monotone variant. In particular, $\text{card-cw}(G)$ ($\text{card-cw}_{\text{mon}}(G)$) is the minimum number of cops necessary to catch the robber in the (monotone) robber and cops game of [ST93].

The next remark is easy to check:

Remark 2.2.10 Let f be a width function on the graph G . Then

$$f\text{-cw}(G) = f^{\text{mon}}\text{-cw}(G).$$

□

The following result will not be used later on, but cf. Proposition 2.4.7 below.

Proposition 2.2.11 Let f be a monotone width function on the finite graph G . Equivalent are:

1. The robber has a positional winning strategy for $\text{RC}(G, f, k)$.
2. There is a function $\xi : \{X \subseteq V(G) \mid f(X) \leq k\} \rightarrow \mathcal{P}(V(G) \setminus \emptyset)$ satisfying
 - (a) $\xi(X) \neq \emptyset$ is a (not necessarily connected) non-empty vertex set of $G \setminus X$,
 - (b) If $X \subseteq Y \subseteq V(G)$ with $f(Y) \leq k$, then $\xi(Y) \subseteq \xi(X)$.
 - (c) If $X \subseteq Y \subseteq V(G)$ with $f(Y) \leq k$, then for every $v \in \xi(X)$ there is a path in $G \setminus X$ to some vertex of $\xi(Y)$.

Proof. $1 \Rightarrow 2$: Assume the robber has a winning strategy against the cops moving on sets $X \subseteq V(G)$ with $f(X) \leq k$. Define

$$\xi(X) := \{r \in V(G) \setminus X \mid \text{the robber can win in position } (X, r)\},$$

for every set $X \subseteq V(G)$ with $f(X) \leq k$. Then ξ satisfies (a) by definition, and (b) holds because if $X \subseteq Y$ with $f(Y) \leq k$ and the robber can win in position (Y, r) then she can win on (X, r) as well.

Finally, for (c) let $X \subseteq Y$ with $f(Y) \leq k$. If the robber can win in position (X, v) and the cops fly to Y , then the robber can run along a path in $G \setminus X$ to some $v' \in V(G)$ such that she can win on (Y, v') .

$2 \Rightarrow 1$: Given ξ , the robber can win as follows: when the cops move to X with $f(X) \leq k$, then she moves to some vertex of $\xi(X)$. It follows from (b) and (c) that this is always possible. □

2.2.3 Monotone strategies

In this section we will prove

Theorem 2.2.12 Let G be a graph, and let f be a width function on G . Then

$$f\text{-hw}(G) = f\text{-cw}_{\text{mon}}(G).$$

As a corollary (using the fact that $\text{tw}(G) + 1 = \text{card} - \text{hw}(G)$) we get a result that was first proved in [ST93]: On a finite graph G , k cops have a monotone winning strategy for the robber and cops game (as defined in [ST93]) iff $\text{tw}(G) + 1 \leq k$. (Of course, the main result of [ST93] was that this holds even without the word ‘monotone’. But in our more general context this is not true.) More special cases will be presented in Chapters 3 and 5.

The rest of this section is devoted to the proof of Theorem 2.2.12. The two directions will follow from Lemmas 2.2.13 and 2.2.18.

From strategies to decompositions

Recall that for a directed tree T and node $t \in T$, t not the root, $\text{pred}(t)$ denotes the unique predecessor of t in T .

Lemma 2.2.13 *Let f be a width function on the graph G , and let $k \in \mathbb{R}$. For every winning strategy σ (for the cops) for $\text{RC}_{\text{mon}}(G, f, k)$ there is a hypertree decomposition (T, B, C) of G with $f\text{-width}(T, B, C) \leq k$.*

Proof. The plays of even length that are consistent with σ form a tree in an obvious way. We will define T as a suitable subtree: For this purpose we fix a choice function γ that associates to every non-empty subset $R \subseteq V(G)$ a vertex $\gamma(R) \in R$.

$V(T)$ is the smallest set of plays of even length that contains the play of length 0 and such that, if $\pi = (X_0, r_0, \dots, X_n, r_n) \in V(T)$, $\sigma(\pi)$ exists and $R \subseteq V(G)$ is a connected component of $G \setminus \sigma(\pi)$ such that R is connected to r_n in $G \setminus (X_n \cap \sigma(\pi))$, then

$$\pi^\frown (\sigma(\pi), \gamma(R)) = (X_0, r_0, \dots, X_n, r_n, \sigma(\pi), \gamma(R)) \in V(T).$$

Note that every $\pi \in T$ is consistent with σ because σ is a winning strategy for the cops, and so $\sigma(\pi)$ exists for every $\pi \in T$. The root of T is the play of length 0. π' is a successor of π if π' is of the form $\pi' = \pi^\frown (X, r)$. We define

$$C_\pi := \sigma(\pi)$$

for all $\pi \in T$. If $\pi = ()$ is the root, we also define $B_{()} := C_{()}$. Otherwise

$$B_\pi := C_\pi \cap (B_{\text{pred}(\pi)} \cup R_\pi).$$

(Recall the definition of R_π , Definition 2.2.2.)

We now show that (T, B, C) is a hypertree decomposition of G .

(TD1): Vertices of G are covered, i. e. are contained in some piece B_π : Otherwise there is a vertex $u \in V(G)$ not covered in any piece of (T, B) . Then the robber can escape by always choosing $\gamma(R)$, where R is a connected component containing u , a contradiction.

(TD2): Edges of G are covered: Suppose $\{u, v\} \in E(G)$. It is not hard to see that there is a tree node π such that $\{u, v\} \subseteq \sigma(\pi)$ and $\{u, v\} \cap R_\pi \neq \emptyset$. (Otherwise the robber could escape by always choosing $\gamma(R)$, where R is a connected component containing u or v .) If π is minimal such (i. e. closest to the root), then also $\{u, v\} \subseteq B_\pi$.

(TD3): For connectedness it is sufficient to show that for every vertex $v \in V(G)$ there is only one tree node such that $v \in B_\pi$ and $v \notin B_{\text{pred}(\pi)}$. It follows from the definition of R_π that the sets $R_{\pi'}$ for children π' of a given node π are disjoint. Hence by (C2') the nodes π such that $v \in R_\pi$ form an initial sequence of a branch of T . Now if $v \in B_\pi$ and $v \notin B_{\pi'}$, then $v \in C_\pi \cap R_\pi$. So π is actually uniquely determined as the last node such that $v \in R_\pi$.

(HD2): By definition, $B_\pi \subseteq C_\pi$ for all $\pi \in T$.

(HD3): If π is the root, then $C_\pi \cap B_{T_\pi} = B_\pi$. Otherwise we claim:

$$B_{T_\pi} \subseteq B_{\text{pred}(\pi)} \cup R_\pi.$$

The claim implies (HD3): $C_\pi \cap B_{T_\pi} = \sigma_\pi \cap B_{T_\pi} \subseteq \sigma_\pi \cap (B_{\text{pred}(\pi)} \cup R_\pi) = B_\pi$.

Proof of the claim. Assume that $v \in B_{T_\pi} \setminus (B_{\text{pred}(\pi)} \cup R_\pi)$ exists. It is not hard to see that for each $u \in V(G)$ there exists exactly one node $\pi_u \in T$ such that $u \in \sigma_{\pi_u} \cap R_{\pi_u}$. Thus $v \in \sigma_{\pi_v} \cap R_{\pi_v} \subseteq B_{\pi_v}$ but $v \notin B_{\text{pred}(\pi)}$. (TD3) implies $\pi_v \in T_\pi$. The monotonicity (C2) of the strategy implies $R_{\pi_v} \subseteq R_\pi$, hence $v \in R_\pi$. This is contrary to the choice of v .

Thus (T, B, C) is a hypertree decomposition for G and since

$$f(C_\pi) = f(\sigma(\pi)) \leq k,$$

we have $f\text{-width}(T, B, C) \leq k$. $\square$

From decompositions to strategies

For this direction, intuitively, the cops move according to the sets C_t of the given hypertree decomposition. They first move to C_w , where w is the root of T . Then the robber chooses a vertex $r \in V(G) \setminus C_w$. Clearly $r \in B_{T_t}$ for exactly one successor t of w . When the cops fly to C_t , the robber cannot leave B_{T_t} during the flight.

The robber chooses a (potentially) new vertex $r' \in V(G) \setminus C_t$. Again, $r' \in B_{T_s}$ for exactly one successor s of t , telling the cops where to move next. Continuing in this way the cops win.

Towards a formalisation of this intuition, we need some definitions and a technical lemma (Lemma 2.2.15).

Definition 2.2.14 Let (T, B) be a tree decomposition of a graph G , and let $t \in T$. We define

$$B_t^{\text{pred}} := \begin{cases} \emptyset & \text{if } t \text{ is the root of } T \\ B_{\text{pred}(t)} & \text{otherwise} \end{cases}$$

The tree component of t is defined as follows:

$$\beta_t := B_{T_t} \setminus B_t^{\text{pred}}.$$

It is a fundamental property of tree decompositions that $B_t \cap B_s$ separates the vertices in the pieces of the two parts of the tree obtained by removing the edge $\{t, s\}$:

Lemma 2.2.15 Let (T, B) be a tree decomposition of the graph G and $\{t, s\} \in E(T)$. Let $X := B_t \cap B_s$. Then every path in G from a vertex $v \in \beta_s$ to a vertex $v' \in V(G) \setminus (X \cup \beta_s)$ has a vertex in X .

Proof. Let u be the last vertex on the path from v to v' such that $u \in \beta_s$. Thus the next vertex u' on the path satisfies $u' \notin \beta_s$. We claim that $u' \in X$:

The graph edge $\{u, u'\}$ is covered by some piece B_r of the tree-decomposition. Since $u \in \beta_s = B_{T_s} \setminus B_t$, u is not covered by any piece of $T \setminus T_s$. (Otherwise (TD3) is not satisfied at t .) Thus $r \in T_s$. On the other hand, since $u' \notin \beta_s$, u' is covered in some piece of $B_{r'}$ of $T \setminus T_s$. Thus $u' \in B_r \cap B_{r'}$ and (TD3) implies that $u' \in B_t \cap B_s = X$. $\square$

Lemma 2.2.16 *Suppose (T, B, C) is a hypertree decomposition of a graph G , $t_0 \in T$, and $R \subseteq B_{T_{t_0}}$ is a connected component of $G \setminus C_{t_0}$. Let t_1 be the root of the subtree (cf. Remark 1.1.3)*

$$\{t \in T \mid B_t \cap R \neq \emptyset\}.$$

Then the following facts hold.

1. $R \subseteq \beta_{t_1}$.
2. $B_{t_1} \cap R \neq \emptyset$.
3. $t_1 \in V(T_{t_0}) \setminus \{t_0\}$.
4. R is a connected component of $G \setminus (C_{t_0} \cap C_{t_1})$.

Proof. 1: This follows from $R \subseteq B_{T_{t_1}}$ and $B_{t_1}^{\text{pred}} \cap R = \emptyset$. 2: This is because $t_1 \in \{t \in T \mid B_t \cap R \neq \emptyset\}$.

3: Note that $C_{t_0} \supseteq B_{t_0}$ by (HD2), so $R \subseteq B_{T_{t_0}} \setminus C_{t_0} \subseteq B_{T_{t_0}} \setminus B_{t_0} \subseteq V(G) \setminus B_{t_0}$. Hence $t_0 \notin \{t \in T \mid B_t \cap R \neq \emptyset\}$. Since, on the other hand, $R \subseteq B_{T_{t_0}}$, we have $t_1 \in \{t \in T \mid B_t \cap R \neq \emptyset\} \subseteq V(T_{t_0})$.

4: Clearly $R \subseteq V(G) \setminus C_{t_0} \subseteq V(G) \setminus (C_{t_0} \cap C_{t_1})$. It remains to show that for every edge $\{r, v\} \in E(G)$ such that $r \in R$ and $v \notin R$ we have $v \in C_{t_0} \cap C_{t_1}$ (and so R is in fact closed under connectivity in $G \setminus (C_{t_0} \cap C_{t_1})$).

Let $s \in T$ be a tree node covering $\{r, v\}$, i. e. $\{r, v\} \subseteq B_s$. Then clearly $s \in \{t \in T \mid B_t \cap R \neq \emptyset\} \subseteq T_{t_1}$. Hence $v \in B_{T_{t_1}}$ (and $v \in B_{T_{t_0}}$ by 3). Since R is closed under connectivity in $G \setminus C_{t_0}$, we have $v \in C_{t_0}$, and $v \in B_{T_{t_0}} \cap C_{t_0} \subseteq B_{t_0}$ by (HD3). Thus $v \in B_{T_{t_1}} \cap B_{t_0}$. Hence by connectedness (TD3) we have $v \in B_{t_1} \subseteq C_{t_1}$. Thus $v \in C_{t_0} \cap C_{t_1}$. $\square$

The following lemma will be used to verify the winning condition.

Lemma 2.2.17 *Let (T, B) be a tree decomposition of a graph G . Then:*

1. $\beta_t \subseteq \beta_{\text{pred}(t)}$, if $t \in T$ is not the root of T .
2. Any infinite branch D of T satisfies $\bigcap_{t \in D} \beta_t = \emptyset$.

Proof. 1:

$$\begin{aligned} \beta_t \setminus \beta_{\text{pred}(t)} &= [B_{T_t} \setminus B_t^{\text{pred}}] \setminus [B_{T_{\text{pred}(t)}} \setminus B_{\text{pred}(t)}^{\text{pred}}] \\ &= (B_{T_t} \setminus [B_{T_{\text{pred}(t)}} \setminus B_{\text{pred}(t)}^{\text{pred}}]) \setminus B_t^{\text{pred}} \\ &= ([B_{T_t} \setminus B_{T_{\text{pred}(t)}}] \cup [B_{T_t} \cap B_{\text{pred}(t)}^{\text{pred}}]) \setminus B_t^{\text{pred}} \\ &= [B_{T_t} \cap B_{\text{pred}(t)}^{\text{pred}}] \setminus B_t^{\text{pred}} \\ &= \emptyset, \end{aligned}$$

where the last equality holds by connectedness (TD3).

2: Suppose $v \in \bigcap_{t \in D} \beta_t = \bigcap_{t \in D} [B_{T_t} \setminus B_t^{\text{pred}}] = \bigcap_{t \in D} B_{T_t} \setminus \bigcup_{t \in D} B_t^{\text{pred}}$. Then all $t \in D$ satisfy $v \notin B_t^{\text{pred}}$. Since D is infinite, this implies that all $t \in D$ satisfy $v \notin B_t$. Since $v \in V(G)$, there is an $s \in T$ such that $v \in B_s$. Let $t \in T$ be the last vertex of D on the path from the root of T to s . Let t' be the successor of t in D . By assumption, we have $v \in B_{t'}$. (TD3) implies that $v \in B_t$, a contradiction. $\square$

Lemma 2.2.18 *Let f be a width function on the graph G , and let $k \in \mathbb{R}$. For every hypertree decomposition (T, B, C) of G with $f\text{-width}(T, B, C) \leq k$ there is a positional winning strategy σ for the cops for $\text{RC}_{\text{mon}}(G, f, k)$.*

Proof. We describe σ informally. The cops' first move is to C_w , where $w \in T$ is the root of T . When the robber moves, resulting in a play $(X_0, r_0, \dots, X_n, r_n)$, then the cops move to a position X_{n+1} determined as follows. Let R be the connected component of r_n in $G \setminus X_n$. Let t be the root of the subtree $\{t \in T \mid B_t \cap R \neq \emptyset\}$ of T . Then $X_{n+1} = C_t$.

The strategy σ for the cops, as defined, is clearly a positional strategy. It remains to show that σ is a winning strategy for $\text{RC}_{\text{mon}}(G, f, k)$. For any play $(X_0, r_0, X_1, \dots)$ consistent with σ let R_n be the connected component of r_n in $G \setminus X_n$ (for $n = 0, 1, \dots$), let $t_0 = w$ be the root of T , and let $t_n \in T$ (for $n = 1, 2, \dots$) be the root of $\{t \in T \mid B_t \cap R_{n-1} \neq \emptyset\}$. Then clearly $X_0 = C_{t_0}$, and $R_0 \subseteq B_{t_0}$ is a connected component of $G \setminus C_{t_0}$. Using Lemma 2.2.16, 4, it is easy to prove by induction that, in fact, for all $n < \omega$ for which the variables are defined we have $X_n = C_{t_n}$, and $R_n \subseteq B_{t_n}$ is a connected component of $G \setminus C_{t_n}$.

Also by Lemma 2.2.16, 4, when the cops move from X_n to X_{n+1} , the robber can only choose a vertex $r_{n+1} \in R_n$, so $R_{n+1} \subseteq R_n$. Hence all cop moves prescribed by σ are actually legal moves in $\text{RC}_{\text{mon}}(G, f, k)$. Therefore the cops win every finite (maximal) play consistent with σ .

For infinite plays note that $R_n \subseteq \beta_{t_{n+1}}$ and $t_{n+1} \in T_{t_n} \setminus \{t_n\}$ (again by Lemma 2.2.16). Therefore $\bigcap_{n < \omega} R_n \subseteq \bigcap_{n < \omega} \beta_{t_{n+1}} = \emptyset$ by Lemma 2.2.17, so the winning condition for infinite plays is also satisfied. $\square$

Corollary 2.2.19 *Let G be a graph and f a width function on G .*

If the cops have a winning strategy for $\text{RC}_{\text{mon}}(G, f, k)$, then they also have a positional winning strategy for $\text{RC}_{\text{mon}}(G, f, k)$.

Proof. By Lemma 2.2.13, a winning strategy for $\text{RC}_{\text{mon}}(G, f, k)$ gives rise to a hypertree decomposition (T, B, C) of G with $f\text{-width}(T, B, C) \leq k$. By Lemma 2.2.18, we obtain a winning strategy for $\text{RC}_{\text{mon}}(G, f, k)$ from (T, B, C) that is actually positional. $\square$

Corollary 2.2.20 *Let f be a width function on the graph G . Then*

$$f\text{-cw}(G) \leq f\text{-ghw}(G).$$

Proof. By Proposition 2.1.21, Theorem 2.2.12, and Remark 2.2.10 we have

$$f\text{-ghw}(G) = f^{\text{mon}}\text{-hw}(G) = f^{\text{mon}}\text{-cw}_{\text{mon}}(G) \geq f^{\text{mon}}\text{-cw}(G) = f\text{-cw}(G).$$

$\square$

2.3 The homomorphism problem

In this section all graphs and structures are finite. The general homomorphism problem asks, given finite relational structures $\mathcal{M}$ and $\mathcal{N}$, whether there exists a homomorphism $\mathcal{M} \xrightarrow{\text{hom}} \mathcal{N}$. For classes $\mathcal{C}$ and $\mathcal{D}$ of finite relational structures, let $\text{HOM}(\mathcal{C}, \mathcal{D})$ denote the restriction of the homomorphism problem to input structures $\mathcal{M} \in \mathcal{C}$ and $\mathcal{N} \in \mathcal{D}$. It is well known that the general homomorphism problem $\text{HOM}(_, _)$ is NP complete⁴. (Here ‘ $_$ ’ denotes the class of all finite relational structures.) Moreover, $\text{HOM}(_, _)$ is equivalent to evaluating conjunctive queries on databases and also to CSP, the Constraint Satisfaction Problem (see [GLS01a]).

We investigate restrictions on the left hand side that lead to polynomial time algorithms, i. e. we look for classes $\mathcal{C}$ such that $\text{HOM}(\mathcal{C}, _) \in \text{P}$.

⁴For the basic notions of complexity theory such as the definitions of the classes P and NP, the reader is referred to [GJ79].

2.3.1 Structures with partial functions

Functional dependencies in databases allow quick computation of the value of a function or a partial function. In Chapter 5 we want to exploit this, so we do not restrict ourselves to relational structures, even though partial functions are of no immediate use in this chapter. We assume that the reader is familiar with the basic notions of model theory (see [EF99]), and we slightly generalise these concepts. This generalisation is in the spirit of ‘partial algebras’ as occasionally examined in universal algebra. (See, e. g., Chapter 2 of [Grä].)

Definition 2.3.1 *A signature is a finite set σ of relation symbols. Some of the relation symbols are marked as partial function symbols. If R is an $(n+1)$ -ary relation symbol that is marked as a partial function symbol, then we may write $R(x_1 \dots x_n) = x_{n+1}$ instead of $Rx_1 \dots x_{n+1}$. For σ -structures we require that the interpretation of a partial function symbol is a partial function on the universe.*

In the above definition, the only difference to the case of general (non-relational) signatures as they are usually considered in model theory is that we have partial functions rather than total functions. As usual, we will distinguish between plain relation symbols and partial function symbols, using capital letters for the former and small letters for the latter. But note that in contrast to the usual treatment of non-relational signatures we will not admit nesting of partial functions in terms. $\exists v(f(x) = v \wedge g(v) = y)$ and $\forall v((f(x) = v) \rightarrow (g(v) = y))$ are (non-equivalent!) formulas in an appropriate signature, but $f(g(x)) = y$ is not unnested, hence not a formula.

Definition 2.3.2

- A substructure of a σ -structure $\mathcal{M}$ is a σ -structure $\mathcal{N}$ such that $R^{\mathcal{N}} = R^{\mathcal{M}} \cap M^n$ for every n -ary relation symbol $R \in \sigma$, and $f^{\mathcal{N}}(\bar{a}) = b$ holds if, and only if, $f^{\mathcal{M}}(\bar{a}) = b$ and $\bar{a}, b \in N$.
- A closed substructure of a σ -structure $\mathcal{M}$ is a substructure $\mathcal{N}$ such that if $f^{\mathcal{M}}(\bar{a}) = b$ holds and $\bar{a} \in N$, then $b \in N$.
- For a subset $X \subseteq M$ of the universe M of $\mathcal{M}$, the structure $\mathcal{M} \upharpoonright X$ is the substructure with universe X .
- For a σ -structure $\mathcal{M}$, the underlying hypergraph $H_{\mathcal{M}}$ of $\mathcal{M}$ is defined as follows.

Hypergraph $H_{\mathcal{M}}$

vertex set: M

edges: $\{a_1, \dots, a_n\}$, where $\mathcal{M} \models Ra_1 \dots a_n$ for some relation symbol $R \in \sigma$
 $\{a_1, \dots, a_{n+1}\}$, where $\mathcal{M} \models p(a_1 \dots a_n) = a_{n+1}$ for some partial function symbol $p \in \sigma$

Observe that for a substructure $\mathcal{M}'$ of $\mathcal{M}$, the underlying hypergraph $H'_{\mathcal{M}}$ of $\mathcal{M}'$ is a subhypergraph of the underlying hypergraph $H_{\mathcal{M}}$ of $\mathcal{M}$, but not in general an induced subhypergraph.

Given a σ -structure $\mathcal{M}$ we distinguish between the cardinality $|M|$ of the universe M of $\mathcal{M}$ and the size $\|\mathcal{M}\|$ of $\mathcal{M}$, given by

$$\|\mathcal{M}\| = |\sigma| + |M| + \sum_{R \in \sigma} |R^{\mathcal{M}}| \cdot \text{arity}(R) + \sum_{p \in \sigma} |p^{\mathcal{M}}| \cdot (\text{arity}(p) - 1).$$

Just like there is more than one natural notion of substructure for signatures with partial function symbols, there are also several notions of homomorphisms between structures in such a signature. In the definition below we choose the weakest, which also happens to be the one we get if we treat σ as a non-relational signature. (Thus if σ' is the result of forgetting the fact that some relation symbols of σ are marked, then the category of σ -structures and homomorphisms is a full subcategory of the category of σ' -structures and homomorphisms.)

Definition 2.3.3 A homomorphism from a σ -structure $\mathcal{M}$ to a σ -structure $\mathcal{N}$ is a mapping $h : M \rightarrow N$ satisfying:

- For all relation symbols $R \in \sigma$ and for all tuples $\bar{a} \in R^{\mathcal{M}}$ we have $h(\bar{a}) \in R^{\mathcal{N}}$.
- For all partial function symbols $p \in \sigma$ and for all tuples $\bar{a}b$ from M with $p^{\mathcal{M}}\bar{a} = b$ we have $p^{\mathcal{N}}(h(\bar{a})) = h(b)$.

We write $\mathcal{M} \xrightarrow{\text{hom}} \mathcal{N}$ to indicate that there exists a homomorphism $h : \mathcal{M} \rightarrow \mathcal{N}$. Given classes $\mathcal{C}$ and $\mathcal{D}$ of finite structures, possibly with partial functions, the homomorphism problem $\text{HOM}(\mathcal{C}, \mathcal{D})$ is the following problem.

$\text{HOM}(\mathcal{C}, \mathcal{D})$
Input: $\mathcal{M} \in \mathcal{C}, \mathcal{N} \in \mathcal{D}$
Question: Is there a homomorphism $h : \mathcal{M} \rightarrow \mathcal{N}$?

We investigate problems of the form $\text{HOM}(\mathcal{C}, _)$, where $\mathcal{C}$ is defined as the class of all structures for which the underlying hypergraph satisfies a certain condition. Such restrictions of the homomorphism problem are often called *structural restrictions*.

2.3.2 A general no-promise algorithm

Definition 2.3.4 For a signature σ and a σ -structure $\mathcal{M}$, the underlying hypergraph $H_{\mathcal{M}}$ of $\mathcal{M}$ is given by:

Hypergraph $H_{\mathcal{M}}$
vertex set: M
edges: $\{a_1, \dots, a_n\}$, where $R^{\mathcal{M}}(a_1, \dots, a_n)$ for some $R \in \sigma$
$\{a_1, \dots, a_{n+1}\}$, where $p^{\mathcal{M}}(a_1 \dots a_n) = a_{n+1}$ for some $p \in \sigma$

Definition 2.3.5 A hypergraph H is acyclic (cf. [LS99]) if its hyperedges can be arranged as nodes of a tree T so that for every vertex $v \in V(H)$, the subgraph of T defined by the nodes containing v is connected.

It is not hard to see that a finite hypergraph H without isolated vertices is acyclic if, and only if, it satisfies $c_H\text{-hw}(H) = 1$ (we will prove this in Proposition 3.4.2). The following is essentially a classical result due to Yannakakis [Ya81]:

Theorem 2.3.6 (Yannakakis 1981) If $\mathcal{C}$ is a class of relational structures such that the underlying hypergraph of each structure in $\mathcal{C}$ is acyclic, then $\text{HOM}(\mathcal{C}, _) \in \text{P}$.

With Proposition 3.4.2, this theorem will be a consequence of our Theorem 2.3.13 below. Nevertheless, as a taste of things to come and in order to avoid forward references, here is a proof.

Proof. Let $\mathcal{M} \in \mathcal{C}$ and $\mathcal{N}$ be an arbitrary structure. We use the following two player game, which is similar to the pebble game defined by Ph. Kolaitis and M. Vardi in [KV95]:

The positions are pairs (S, h) , where $S \in E(H_{\mathcal{M}})$ and h is a homomorphism $h : \mathcal{M} \upharpoonright S \rightarrow \mathcal{N}$.

In each round, player I chooses a new subset $S \in E(H_{\mathcal{M}})$. Then player II chooses a new homomorphism $h : \mathcal{M} \upharpoonright S \rightarrow \mathcal{N}$ subject to the following *compatibility condition*: If (S', h') was the previous position, then $h(a) = h'(a)$ for all $a \in S \cap S'$. (In the first round the compatibility condition makes no sense, and player II may choose an arbitrary homomorphism $h : \mathcal{M} \upharpoonright S \rightarrow \mathcal{N}$. Equivalently, one could say that the game starts from the initial position $(\emptyset, \emptyset)$, even though this is not a legal position unless $\emptyset \in E(H_{\mathcal{M}})$.)

Player I wins the game, if player II cannot move. Player II wins all infinite games.

Claim. There is a homomorphism $\mathcal{M} \rightarrow \mathcal{N}$ if and only if player II has a winning strategy for the game.

The forward direction is obvious. For the other direction, suppose player II has a winning strategy. Let T be a tree with $V(T) = E(H_{\mathcal{M}})$ witnessing that $H_{\mathcal{M}}$ is acyclic. Since player II can win, he can also win if player I chooses the subsets $S \subseteq M$ according to a branch of T , starting with $e_r \in E(H_{\mathcal{M}})$ where e_r is the root of T , and ending with $e_\ell \in E(H_{\mathcal{M}})$ where e_ℓ is a leaf of T . Thus for every $e \in T$ we obtain a homomorphism h_e such that $h_e : \mathcal{M} \upharpoonright e \xrightarrow{\text{hom}} \mathcal{N}$. Because for each vertex $a \in V(H_{\mathcal{M}})$ the subgraph of T defined by the nodes containing a is connected, with the compatibility condition we obtain a homomorphism $h : \mathcal{M} \rightarrow \mathcal{N}$ by setting $h(a) := h_e(a)$ for a node $e \in T$ with $a \in e$.

There is a straightforward dynamic programming algorithm deciding in polynomial time whether player I has a winning strategy. $\square$

It turns out that it is sufficient to impose our structural restrictions on certain substructures of the elements in $\mathcal{C}$:

Definition 2.3.7 *The core of a structure $\mathcal{M}$ is the smallest substructure $\mathcal{M}' \subseteq \mathcal{M}$ such that there is a homomorphism $\mathcal{M} \xrightarrow{\text{hom}} \mathcal{M}'$.*

Remark 2.3.8 *Let $\mathcal{M}'$ be a core of $\mathcal{M}$.*

1. *For a structure $\mathcal{N}$ there is a homomorphism $\mathcal{M} \xrightarrow{\text{hom}} \mathcal{N}$ if and only if there is a homomorphism $\mathcal{M}' \xrightarrow{\text{hom}} \mathcal{N}$.*
2. *Any two cores of $\mathcal{M}$ are isomorphic, so the core is well-defined up to isomorphism.*

We will generalise the following result of Dalmau, Kolaitis and Vardi [DKV02]:

Fact 2.3.9 (Dalmau, Kolaitis, Vardi) *For an integer $k \geq 1$ and a class $\mathcal{C}$ of structures such that the core of each structure in $\mathcal{C}$ has tree-width at most k , the problem $\text{HOM}(\mathcal{C}, \perp)$ is in P.*

Intuitively, instead of admitting only structures with cores of tree-width at most k , we will admit classes of structures with cores that admit generalised hypertree decompositions with guards in some predescribed class $\mathcal{C}_1$. For our purposes we need the class $\mathcal{C}_1$ to be such that the following two problems are solvable in polynomial time.

$\text{SUBSTR}_{\text{enum}}^{\mathcal{C}_1}$
Input: A structure $\mathcal{M}$ Output: All closed substructures of $\mathcal{M}$ that are in $\mathcal{C}_1$.

$\text{HOM}_{\text{enum}}(\mathcal{C}_1, -)$
Input: Structures $\mathcal{M} \in \mathcal{C}_1, \mathcal{N}$ Output: All homomorphisms $h : \mathcal{M} \rightarrow \mathcal{N}$.

Of course, if $\mathcal{C}_1$ is the class of all finite structures, these problems are not in P.

Definition 2.3.10 Let $\mathcal{C}_1$ be a class of structures, and let $\mathcal{M}$ be a structure. We say that (T, B, C) is a (generalised) $\mathcal{C}_1$ -hypertree decomposition of $\mathcal{M}$, if

- (T, B, C) is a (generalised) hypertree decomposition of $H_{\mathcal{M}}$,
- $\mathcal{M} \upharpoonright C_t$ is a closed substructure of $\mathcal{M}$, and
- $\mathcal{M} \upharpoonright C_t \in \mathcal{C}_1$ for all $t \in T$.

For a class $\mathcal{C}_1$ and for a structure $\mathcal{M}$ we define the width function

$$f_{\mathcal{C}_1} : \mathcal{P}(V(H_{\mathcal{M}})) \rightarrow \mathbb{R} \cup \{\infty\},$$

$$X \mapsto \begin{cases} 1 & \text{if } \mathcal{M} \upharpoonright X \text{ is a closed substructure of } \mathcal{M} \text{ in } \mathcal{C}_1, \\ \infty & \text{otherwise.} \end{cases}$$

Remark 2.3.11 Let $\mathcal{C}_1$ be a class of structures. The structure $\mathcal{M}$ has a (generalised) $\mathcal{C}_1$ -hypertree decomposition if, and only if, $H_{\mathcal{M}}$ has a (generalised) hypertree decomposition of $f_{\mathcal{C}_1}$ -width 1.

Conversely, let f be a family of width functions f^H , for all hypergraphs H . For example, $f^H = \text{card}$, or $f^H = c_H$. For $k > 0$ put $\mathcal{C}_k^f := \{\mathcal{M} \mid f^{H_{\mathcal{M}}}(H_{\mathcal{M}}) \leq k\}$.

For a hypergraph $H = (V, E)$ and $X \subseteq V$ define $H \upharpoonright X := (X, E \cap \mathcal{P}(X))$. Note that $H_{\mathcal{M}} \upharpoonright X = H_{\mathcal{M} \upharpoonright X}$ for a structure $\mathcal{M}$ and $X \subseteq M$.

Remark 2.3.12 Suppose f is a family of width functions f^H such that for every hypergraph $H = (V, E)$ and every $X \subseteq V$ we have $f^H(X) = f^{H \upharpoonright X}(X)$. Let $\mathcal{M}$ be a relational structure. Then (T, B, C) is a (generalised) $\mathcal{C}_k^f$ -hypertree decomposition of $\mathcal{M}$ if, and only if (T, B, C) is a (generalised) hypertree decomposition of $H_{\mathcal{M}}$ of $f^{H_{\mathcal{M}}}$ -width at most k .

Proof. Let (T, B, C) be a (generalised) $\mathcal{C}_k^f$ -hypertree decomposition of $\mathcal{M}$. Then $\mathcal{M} \upharpoonright C_t \in \mathcal{C}_k^f$ for all $t \in T$. Thus $f^{H_{\mathcal{M} \upharpoonright C_t}}(C_t) \leq k$ for all $t \in T$. By assumption, for every $t \in T$ we have $f^{H_{\mathcal{M}}}(C_t) = f^{H_{\mathcal{M} \upharpoonright C_t}}(C_t) \leq k$. Thus, (T, B, C) is a (generalised) hypertree decomposition of $H_{\mathcal{M}}$ of $f^{H_{\mathcal{M}}}$ -width at most k .

Conversely, let (T, B, C) be a (generalised) hypertree decomposition of $H_{\mathcal{M}}$ of $f^{H_{\mathcal{M}}}$ -width at most k . Then for every $t \in T$ we have $f^{H_{\mathcal{M}}}(C_t) = f^{H_{\mathcal{M} \upharpoonright C_t}}(C_t) \leq k$. Thus (T, B, C) is a (generalised) $\mathcal{C}_k^f$ -hypertree decomposition of $\mathcal{M}$. $\square$

Note that for $f^H = \text{card}$ and for $f^H = c_H$ we have $f^H(X) = f^{H \upharpoonright X}(X)$. (This is not true for $f^H = c_H^{\text{mon}}$.)

For a class $\mathcal{C}_1$ of structures, let

$$\text{COREDECOMPOSABLE}_{\text{ghd}}^{\mathcal{C}_1} := \{\mathcal{M} \mid \text{the underlying hypergraph } H_{\mathcal{M}'} \text{ of the core } \mathcal{M}' \text{ of } \mathcal{M} \text{ has a generalised } \mathcal{C}_1\text{-hypertree decomposition}\}.$$

Theorem 2.3.13 *Let $\mathcal{C}_1$ be a class of finite structures such that*

- $\text{SUBSTR}_{\text{enum}}^{\mathcal{C}_1} \in \text{P}$, and
- $\text{HOM}_{\text{enum}}(\mathcal{C}_1, -) \in \text{P}$.

Then $\text{HOM}(\text{COREDECOMPOSABLE}_{\text{ghd}}^{\mathcal{C}_1}, -) \in \text{P}$.

Proof. Let $\mathcal{M} \in \text{COREDECOMPOSABLE}_{\text{ghd}}^{\mathcal{C}_1}$ and $\mathcal{N}$ an arbitrary structure. The following is a generalisation of the *pebble game* from [KV95]:

The positions are pairs (S, h) where $S \subseteq M$ is such that $\mathcal{M} \upharpoonright S \in \mathcal{C}_1$ is a closed substructure, and h is a homomorphism $h : \mathcal{M} \upharpoonright S \rightarrow \mathcal{N}$ of σ -structures. In each round, player I chooses a new subset $S \subseteq M$ with $\mathcal{M} \upharpoonright S \in \mathcal{C}_1$. Then player II chooses a new homomorphism $h : \mathcal{M} \upharpoonright S \rightarrow \mathcal{N}$ subject to the following *compatibility condition*: If (S', h') was the previous position, then $h(a) = h'(a)$ for all $a \in S \cap S'$.

Player I wins if player II cannot move. Player II wins all infinite games.

Claim. There is a homomorphism $\mathcal{M} \rightarrow \mathcal{N}$ if and only if player II has a winning strategy for the game.

The forward direction is obvious. For the other direction, suppose player II has a winning strategy. By Remark 2.3.8 it suffices to show that there is a homomorphism from the core $\mathcal{M}'$ of $\mathcal{M}$ to $\mathcal{N}$. Let (T, B, C) be a generalised $\mathcal{C}_1$ -hypertree decomposition of $\mathcal{M}'$. Since player II can win, he can also win if player I chooses the subsets $S \subseteq M$ according to a branch of T , starting with C_r for the root $r \in T$, and ending with C_ℓ for a leaf $\ell \in T$. Thus for every node $t \in T$ we obtain a homomorphism h_t such that $h_t : \mathcal{M}' \upharpoonright C_t \xrightarrow{\text{hom}} \mathcal{N}$. In particular, for every node $t \in T$ the mapping $h'_t := h_t \upharpoonright B_t$ is a homomorphism from the substructure $\mathcal{M}' \upharpoonright B_t$ of $\mathcal{M}'$ to $\mathcal{N}$. Because of connectedness (TD3) and the compatibility condition we obtain a homomorphism $h : \mathcal{M}' \rightarrow \mathcal{N}$ by setting $h(a) := h'_t(a)$ for a node $t \in T$ with $a \in B_t$.

There is a straightforward dynamic programming algorithm deciding whether player I has a winning strategy. Since by assumption the problems $\text{SUBSTR}_{\text{enum}}^{\mathcal{C}_1}$ and $\text{HOM}_{\text{enum}}(\mathcal{C}_1, -)$ are in P, the algorithm is in P as well. $\square$

Corollary 2.3.14 *Fix an integer $k > 0$.*

Then the class $\mathcal{C}_k^{\text{card}}$ satisfies the assumptions of Theorem 2.3.13, and thus

$$\text{HOM}(\text{COREDECOMPOSABLE}_{\text{ghd}}^{\mathcal{C}_k^{\text{card}}}, -) \in \text{P}.$$

$\square$

Thus for a class $\mathcal{C}_1$, such that the core $\mathcal{M}'$ of each structure $\mathcal{M} \in \mathcal{C}_1$ satisfies $\text{card-ghw}(H_{\mathcal{M}'}) \leq k$, (i.e. $\text{tw}(H_{\mathcal{M}'}) + 1 \leq k$), the problem $\text{HOM}(\mathcal{C}_1, -)$ is in P. This reproves Fact 2.3.9. Similarly, if the core $\mathcal{M}'$ of each structure $\mathcal{M} \in \mathcal{C}_1$ satisfies $c_{H_{\mathcal{M}'}}\text{-ghw}(H_{\mathcal{M}'}) \leq k$.

In [Gr03], M. Grohe proved the following beautiful classification theorem.

Fact 2.3.15 (Grohe 2003) *Assume that $\text{FPT} \neq \text{W}[1]$.⁵ Then for every recursively enumerable class $\mathcal{C}$ of relational structures of bounded arity the following statements are equivalent:*

⁵See [DF99] for the definitions of FPT and W[1].

- $\text{HOM}(\mathcal{C}, _) \in \text{P}$
- *There is an integer $k \geq 1$ such that the core of every structure in $\mathcal{C}$ has tree-width at most k .*

Thus essentially, for relational structures of bounded arity, bounded tree-width of the core is the best we can get. For classes of structures of unbounded arity this is not the case, and the problem of characterising the tractable classes of unbounded arity is still open.

The algorithm described in the proof of Theorem 2.3.13 is a ‘no promise algorithm’ in the following sense: If we apply the algorithm to structures $\mathcal{M}, \mathcal{N}$ where $\mathcal{M} \notin \text{COREDECOMPOSABLE}_{\text{ghd}}^{\mathcal{C}_1}$, then there is no promise that the algorithm will detect this. (In fact, it is likely to give a wrong answer.)

2.3.3 A general promise algorithm

In [GLS01a], the authors claim that an appropriate notion of hypergraph width should fulfil both of the following conditions:

1. Relevant hypergraph-based problems should be solvable in polynomial time for instances of bounded width.
2. For each constant k , one should be able to check in polynomial time whether a hypergraph is of width k , and, in the positive case, it should be possible to produce an associated decomposition of width k in polynomial time.

We show that this is the case for $\mathcal{C}_1$ -hypertree decompositions, provided that the problems $\text{SUBSTR}_{\text{enum}}^{\mathcal{C}_1}$ and $\text{HOM}_{\text{enum}}(\mathcal{C}_1, _)$ are in P. For a class $\mathcal{C}_1$ of structures, let

$$\text{DECOMPOSABLE}_{\text{hd}}^{\mathcal{C}_1} := \{\mathcal{M} \mid H_{\mathcal{M}} \text{ has a } \mathcal{C}_1\text{-hypertree decomposition}\}.$$

$\text{HD}^{\mathcal{C}_1}$
Input: A structure $\mathcal{M}$ Output: A $\mathcal{C}_1$ -hypertree decomposition (T, B, C) of $H_{\mathcal{M}}$ if $\mathcal{M} \in \text{DECOMPOSABLE}_{\text{hd}}^{\mathcal{C}_1}$, ‘failure’ otherwise.

Theorem 2.3.16 *Let $\mathcal{C}_1$ be a class of finite structures such that*

$$\text{SUBSTR}_{\text{enum}}^{\mathcal{C}_1} \in \text{P}.$$

Then

$$\text{HD}^{\mathcal{C}_1} \in \text{P}.$$

Proof sketch. Given a structure $\mathcal{M}$, by Remark 2.3.11 and the game theoretic characterisation, $\mathcal{M}$ has a $\mathcal{C}_1$ -hypertree decomposition if, and only if, the cops have a winning strategy for $\text{RC}_{\text{mon}}(H_{\mathcal{M}}, f_{\mathcal{C}_1}, 1)$. There is a straightforward dynamic programming algorithm, that decides whether the cops have a winning strategy: List all pairs (X, r) with $X \subseteq M$, $\mathcal{M} \upharpoonright X \in \mathcal{C}_1$, and $r \in M$. By assumption, this is possible in polynomial time. Then mark all pairs (X, r) with $r \in X$ as *winning positions*. Given the list of all pairs, some of which are marked as winning positions, we mark a pair (X', r') as a winning position if there is a set $X \subseteq M$ satisfying:

- The connected component R' of $H_{\mathcal{M}} \setminus X'$ containing r' is the component of $H_{\mathcal{M}} \setminus (X' \cap X)$ that contains r' (monotonicity),

- For all vertices $r \in M$ that are connected to r' in $H_{\mathcal{M}} \setminus (X' \cap X)$, the pair (X, r) is marked as a winning position.

Since the strategy must be monotone, after $|M|$ iterations we can check whether all initial positions $(\emptyset, r)$ with $r \in M$ are marked as winning positions. In this case, the cops' winning strategy can be transformed into a $\mathcal{C}_1$ -hypertree decomposition. Then return the $\mathcal{C}_1$ -hypertree decomposition. Otherwise return 'failure'. $\square$

Thus, if $\text{SUBSTR}_{\text{enum}}^{\mathcal{C}_1} \in \text{P}$, then $\mathcal{C}_1$ -hypertree decompositions satisfy Claim 2 from [GLS01a] cited above.

By assembling Theorem 2.3.16 with Theorem 2.3.13 it is easy to get a 'promise algorithm' for $\text{HOM}(\text{DECOMPOSABLE}_{\text{hd}}^{\mathcal{C}_1}, -)$:

Corollary 2.3.17 *Let $\mathcal{C}_1$ be a class of structures such that*

- $\text{SUBSTR}_{\text{enum}}^{\mathcal{C}_1} \in \text{P}$, *and*
- $\text{HOM}_{\text{enum}}(\mathcal{C}_1, -) \in \text{P}$.

Then there is an algorithm that decides, for given input $\mathcal{M}, \mathcal{N}$, whether $H_{\mathcal{M}}$ has a $\mathcal{C}_1$ -hypertree decomposition. If so, then it correctly answers the question whether there is a homomorphism $h : \mathcal{M} \rightarrow \mathcal{N}$. If not, it rejects.

Dropping the condition $\text{SUBSTR}_{\text{enum}}^{\mathcal{C}_1} \in \text{P}$, the homomorphism problem remains solvable in polynomial time, provided that the left hand side input comes equipped with a generalised $\mathcal{C}_1$ -decomposition. Towards this we define:

$$\text{DECOMPOSED}^{\mathcal{C}_1} := \{ (\mathcal{M}, (T, B, C)) \mid (T, B, C) \text{ is a generalised } \mathcal{C}_1\text{-hypertree decomposition of } H_{\mathcal{M}} \}.$$

$\text{HOM}(\text{DECOMPOSED}^{\mathcal{C}_1}, -)$
Input: Structures $\mathcal{M}, \mathcal{N}$ and a generalised $\mathcal{C}_1$ -hypertree decomposition (T, B, C) von $H_{\mathcal{M}}$.
Question: Is there a homomorphism $h : \mathcal{M} \rightarrow \mathcal{N}$?

Theorem 2.3.18 *Let $\mathcal{C}_1$ be a class of structures such that*

$$\text{HOM}_{\text{enum}}(\mathcal{C}_1, -) \in \text{P}.$$

Then

$$\text{HOM}(\text{DECOMPOSED}^{\mathcal{C}_1}, -) \in \text{P}.$$

Proof. Roughly, the idea is as follows: Given σ -structures $\mathcal{M}$ and $\mathcal{N}$ and a generalised $\mathcal{C}_1$ -hypertree decomposition (T, B, C) of $\mathcal{M}$, we define σ' and σ' -structures $\mathcal{M}'$ and $\mathcal{N}'$ such that

- $M' = M, N' = N$,
- $H_{\mathcal{M}'}$ is *acyclic*,
- h is a homomorphism from $\mathcal{M}$ to $\mathcal{N}$ if, and only if, h is a homomorphism from $\mathcal{M}'$ to $\mathcal{N}'$.

Then we use Yannakakis' polynomial algorithm (Theorem 2.3.6) for deciding whether there is a homomorphism $h : \mathcal{M}' \rightarrow \mathcal{N}'$. More formally, we define σ' and the σ' -structures $\mathcal{M}'$ and $\mathcal{N}'$ as follows:

- $\sigma' := \{R_t \mid t \in T\}$, where R_t has arity $|B_t|$.
- $M' := M$, $N' := N$.
- For an enumeration $a_1, \dots, a_n$ of $B_t \subseteq M$, let
 $R_t^{\mathcal{M}'} := \{(a_1, \dots, a_n)\}$, and
 $R_t^{\mathcal{N}'} := \{(h(a_1), \dots, h(a_n)) \mid h : \mathcal{M} \upharpoonright C_t \xrightarrow{\text{hom}} \mathcal{N}\}$

$\mathcal{M}'$ and $\mathcal{N}'$ can be constructed in polynomial time by assumption. Note that $H_{\mathcal{M}'}$ has hyperedges $\{B_t \mid t \in T\}$, thus $H_{\mathcal{M}'}$ is acyclic.

Claim. $h : \mathcal{M} \xrightarrow{\text{hom}} \mathcal{N}$ if, and only if, $h : \mathcal{M}' \xrightarrow{\text{hom}} \mathcal{N}'$.

Proof of the claim. The direction from left to right follows from the definitions. Conversely, for a partial function symbol $p \in \sigma$, suppose $\mathcal{M} \models p(a_1, \dots, a_n) = a_{n+1}$. Then

$$\{a_1, \dots, a_{n+1}\} \in E(H_{\mathcal{M}}),$$

and thus $\{a_1, \dots, a_{n+1}\} \subseteq B_t$ for some $t \in T$, because every hyperedge is covered in some piece of T . In particular, $h \upharpoonright B_t : \mathcal{M}' \upharpoonright B_t \xrightarrow{\text{hom}} \mathcal{N}'$, and thus by definition $h \upharpoonright B_t : \mathcal{M} \upharpoonright B_t \xrightarrow{\text{hom}} \mathcal{N}$. Therefore $\mathcal{N} \models p(h(a_1), \dots, h(a_n)) = h(a_{n+1})$.

For a relation symbol $R \in \sigma$ the proof is similar. $\square$

Thus, if $\text{HOM}_{\text{enum}}(\mathcal{C}_1, _) \in \text{P}$, (generalised) $\mathcal{C}_1$ -hypertree decompositions satisfy Claim 1 from [GLS01a] cited above.

Let $\mathcal{C}_1$ be a class of structures, and let $\mathcal{M}$ be a structure. Recall that $H_{\mathcal{M}}$ has a $\mathcal{C}_1$ -hypertree decomposition if, and only if, the cops have a winning strategy for the game $\text{RC}_{\text{mon}}(H_{\mathcal{M}}, f_{\mathcal{C}_1}, 1)$. There is an analogous theorem to 2.3.16 for the non-monotone game. Let $\text{CS}^{\mathcal{C}_1}$ be the following problem (where CS stands for cop strategy):

$\text{CS}^{\mathcal{C}_1}$
Input: A structure $\mathcal{M}$
Output: A winning strategy for the cops in $\text{RC}(H_{\mathcal{M}}, f_{\mathcal{C}_1}, 1)$, if there is one, 'failure', otherwise.

The following is a corollary to the proof of Theorem 2.3.16.

Corollary 2.3.19 *If $\mathcal{C}_1$ is a class of finite structures such that*

$$\text{SUBSTR}_{\text{enum}}^{\mathcal{C}_1} \in \text{P}.$$

Then

$$\text{CS}^{\mathcal{C}_1} \in \text{P}.$$

Proof sketch. The proof is similar to the proof of Theorem 2.3.16. Let $\mathcal{M}$ be an arbitrary structure. There is a straightforward dynamic programming algorithm deciding whether the cops have a winning strategy:

Again, list all pairs (X, r) with $X \subseteq M$, $\mathcal{M} \upharpoonright X \in \mathcal{C}_1$, and $r \in M$. By assumption, this is possible in polynomial time. Then mark all pairs (X, r) with $r \in X$ as *winning positions*. Given the list of all pairs some of which are marked as winning positions, we mark a pair (X', r') as a winning positions, if there is a set $X \subseteq M$ satisfying:

- For all vertices $r \in M$ that are connected to r' in $H_{\mathcal{M}} \setminus (X' \cap X)$, the pair (X, r) is marked as a winning position.

Let p be a polynomial witnessing $\text{SUBSTR}_{\text{enum}}^{C_1} \in \text{P}$. Thus there are at most $p(\|\mathcal{M}\|) \cdot |M|$ different positions. Thus, after $p(\|\mathcal{M}\|) \cdot |M|$ iterations we can check whether all initial positions $(\emptyset, r)$ with $r \in M$ are marked as winning positions. If not, return ‘failure’, otherwise return a winning strategy. $\square$

2.3.4 Conjunctive query evaluation

We say that a σ -formula φ is a *conjunctive query* if φ is a conjunction of positive σ -atoms.

For relational signatures, there is a well known translation between the homomorphism problem and conjunctive query evaluation. We will show that this translation also works for non-relational structures. But before doing so, we will show how conjunctive queries and structures are related to queries on real databases.

Conjunctive queries and databases

The problem of conjunctive query evaluation is the following problem:

CQ
Input: A structure $\mathcal{N}$ and a conjunctive query φ
Question: $\varphi(\mathcal{N}) \neq \emptyset$?
(I. e.: Are there elements $a_1, \dots, a_n \in N$ such that $\mathcal{N} \models \varphi(a_1, \dots, a_n)$?)

Here $\mathcal{N}$ is the ‘database’ (possibly with functional dependencies). For example consider the following database, consisting of three tables:

employee_data:	id	name	year_of_birth
	1	Tsatsikakis	1968
	2	Souvlakopoulou	1976
	3	Dolmadaki	1942
	4	Giaourtimelidis	1976

employee_salaries:	id	salary_group
	1	II
	2	III
	3	I
	4	II

salary_details:	salary_group	year_of_birth	salary	new_salary
	I	1942	70000	70000
	I	1968	65000	64000
	I	1976	60000	58000
	II	1942	55000	52000
	II	1968	50000	47000
	II	1976	45000	42000
	III	1942	42000	39000
	III	1968	40000	37000
	III	1976	38000	35000

Ignoring the double lines for the moment, this database can be regarded as a relational structure $\mathcal{N}$ in the signature with relation symbols $R_{\text{employee_data}}$ (ternary),

$R_{\text{employee_salaries}}$ (binary) and $R_{\text{salary_details}}$ (4-ary) in an obvious way. Thus the elements of the structure are 1, 2, 3, 4, Tsatsikakis, Souvlakopoulou, Dolmadaki, Giaourtimelidis, I, II, III, 1942, 1968, 1976, 70000, 64000,

A query one might ask is: Is there an employee whose salary won't change? In SQL we could code this as follows:

```
SELECT * FROM employee_data, employee_details, salary_details
WHERE employee_data.id=employee_details.id
  AND employee_salaries.salary_group=salary_details.salary_group
  AND employee_data.year_of_birth=salary_details.year_of_birth
  AND salary_details.salary=salary_details.new_salary
LIMIT 1;
```

As a conjunctive query this can be formalised as follows:

$$\begin{aligned} \varphi(x_{\text{id}}, x_{\text{name}}, x_{\text{yob}}, x_{\text{sg}}, x_{\text{salary}}) = & R_{\text{employee_data}}(x_{\text{id}}, x_{\text{name}}, x_{\text{yob}}) \\ & \wedge R_{\text{employee_salaries}}(x_{\text{id}}, x_{\text{sg}}) \\ & \wedge R_{\text{salary_details}}(x_{\text{sg}}, x_{\text{yob}}, x_{\text{salary}}, x_{\text{salary}}). \end{aligned}$$

Then the question is whether $\varphi(\mathcal{N}) \neq \emptyset$.

So far we have ignored the double lines in the tables, which indicate that the combined columns before the double line together form the key of the table. Thus in the `salary_details` table, for example, for every combination of potential values in the first two columns there is at most one line matching these values. This restriction on the database can of course be used for faster evaluation of queries. We can translate it into the structure $\mathcal{N}$ by adding unary partial functions $f_{\text{employee_data.name}}$, $f_{\text{employee_data.year_of_birth}}$, $f_{\text{employee_salaries.salary_group}}$ and binary partial functions $f_{\text{salary_details.salary}}$ and $f_{\text{salary_details.new_salary}}$.

Conjunctive queries and the homomorphism problem

Let $\mathcal{M}$ and $\mathcal{N}$ be σ -structures. It is easy to see that

- $\mathcal{M} \xrightarrow{\text{hom}} \mathcal{N} \iff \varphi_{\mathcal{M}}(\mathcal{N}) \neq \emptyset$,

where

$$\varphi_{\mathcal{M}} = \bigwedge_{\substack{R \in \sigma \\ \mathcal{M} \models R\bar{a}}} R\bar{a} \quad \wedge \quad \bigwedge_{\substack{p \in \sigma \\ \mathcal{M} \models p(\bar{a})=b}} p(\bar{a}) = b.$$

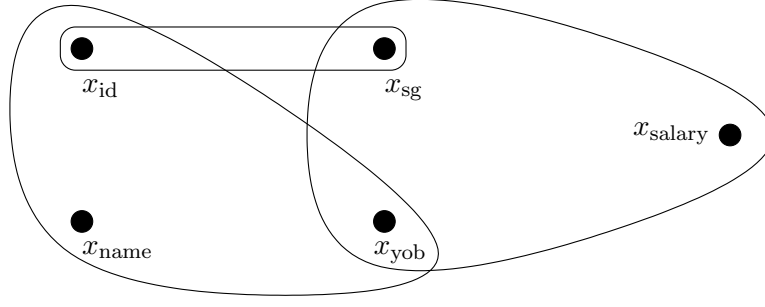
- For a conjunctive query φ we have $\varphi(\mathcal{N}) \neq \emptyset \iff \mathcal{M}_{\varphi} \xrightarrow{\text{hom}} \mathcal{N}$,

where

$M_{\varphi} = \text{var}(\varphi) / \sim$ is the set of variables of φ up to the equivalence relation $\sim$, with $x \sim y$ if $\varphi \models (x = y)$,
 $R^{\mathcal{M}}\bar{x}$ holds if $\varphi \models R\bar{x}$, and $p^{\mathcal{M}}\bar{x} = y$ holds if $\varphi \models (p\bar{x} = y)$.

We can apply this to our example query $\varphi = \varphi(x_{\text{id}}, x_{\text{name}}, x_{\text{yob}}, x_{\text{sg}}, x_{\text{salary}})$. As a homomorphism problem, the question is, whether there is a homomorphism $\mathcal{M}_{\varphi} \xrightarrow{\text{hom}} \mathcal{N}$. Figure 2.3 shows the *underlying hypergraph* H_{φ} of φ , which is defined as the underlying hypergraph of $\mathcal{M}_{\varphi}$. Is easy to see that $c_{H_{\varphi}}\text{-hw}(H_{\varphi}) = 2$.

We could also have coded our example query in a more efficient way, using the functional dependencies:

Figure 2.3: The underlying hypergraph H_φ of the example query.

$$\begin{aligned}
\psi(x_{\text{id}}, x_{\text{yob}}, x_{\text{sg}}, x_{\text{salary}}) = & f_{\text{employee_data.year_of_birth}}(x_{\text{id}}) = x_{\text{yob}} \\
& \wedge f_{\text{employee_salaries.salary_group}}(x_{\text{id}}) = x_{\text{sg}} \\
& \wedge f_{\text{salary_details.salary}}(x_{\text{sg}}, x_{\text{yob}}) = x_{\text{salary}} \\
& \wedge f_{\text{salary_details.new_salary}}(x_{\text{sg}}, x_{\text{yob}}) = x_{\text{salary}}.
\end{aligned}$$

It is easy to check that $c_{H_\psi}\text{-hw}(H_\psi) = 2$, so the increased efficiency due to the fact that we need only iterate over all possible values of x_{id} is not captured in the underlying hypergraph. In Chapter 5 we will define the underlying directed hypergraph $\vec{H}_\psi$ of $\mathcal{M}_\psi$ and a corresponding notion of hypertree-width. With these improved notions we will have $\vec{c}_{\vec{H}_\psi}\text{-hw}(\vec{H}_\psi) = 1$.

2.4 Related f -invariants

In this section we are mainly interested in monotone width functions.

We define some more invariants for equipped graphs that resemble graph invariants such as the *bramble number* or *tangle number* of a graph [Re97, RS91], the *branch-width* [RS91], the *linkedness* [Re97] and we prove that for certain classes of equipped graphs all our invariants are linearly coherent in the following sense:

Definition 2.4.1 *Let I, J be two invariants defined on a class $\mathcal{C}$ of graphs equipped with a width function.*

- *I and J are coherent [Ha76] on $\mathcal{C}$, if there exist functions $g, h : \mathbb{R} \rightarrow \mathbb{R}$, such that*

$$I(G, f) \leq r \implies J(G, f) \leq g(r), \text{ and } J(G, f) \leq r \implies I(G, f) \leq h(r)$$

for all $(G, f) \in \mathcal{C}$.

- *I and J are linearly coherent on $\mathcal{C}$, if they are coherent and g, h can be chosen to be linear functions.*
- *I and J are strongly coherent on $\mathcal{C}$, if there exist $c, C \in \mathbb{R}$, such that*

$$J(G, f) \leq I(G, f) + c \text{ and } I(G, f) \leq J(G, f) + C$$

for all $(G, f) \in \mathcal{C}$.

The graph invariants mentioned above are known to be linearly coherent in this sense with tree-width.

The purpose of this section is, on one hand, to prove that under some conditions on f , we have $f\text{-ghw}(G) \leq 3 \cdot f\text{-cw}(G) + 2$ (as a consequence of Theorem 2.4.23, 1 and 3), hence linear coherence of $f\text{-cw}$ and $f\text{-ghw}$. On the other hand, we generalise some graph invariants by introducing $f\text{-bramble-number}$, $f\text{-tangle-number}$, $f\text{-branch-width}$, and $f\text{-linkedness}$, and we show that they are linearly coherent on a certain class of equipped graphs. Outside this section we will not refer to these invariants beyond an occasional corollary to Theorem 2.4.23.

This section is in part based on joint work with G. Gottlob and M. Grohe [AGG05].

2.4.1 Conditions on width functions

For a graph G we say that two sets $X, Y \subseteq V(G)$ *touch*, if $X \cap Y \neq \emptyset$ or there is an edge $e \in E(G)$ with $e \cap X \neq \emptyset$ and $e \cap Y \neq \emptyset$.

Definition 2.4.2 (Conditions on width functions) *Let f be a width function on a graph G .*

- f is weakly submodular, if any two finite subsets $X, Y \subseteq V(G)$ satisfy

$$f(X \cup Y) \leq f(X) + f(Y).$$

- f is additive, if any two finite sets $X, Y \subseteq V(G)$ that do not touch satisfy

$$f(X \cup Y) = f(X) + f(Y).$$

- f is tame if for all $v \in V(G)$ there is a finite set $X \subseteq V(G)$ such that $v \in X$ and $f(X) \leq 1$.

for all vertices $v \in V(G)$.

Clearly, a hypergraph H is tame if, and only if, c_H is a tame width function on $\underline{H}$.

It should perhaps be noted, that weak submodularity and tameness are properties of f only, while additivity is a property of both f and the graph: If f is a weakly submodular (or tame) width function on a graph G , and G' is a graph $V(G') = V(G)$, then f is also a weakly submodular (tame, resp.) width function on G' . The analogous statement for additivity is false: Take $G = K_2$ together with the width function

$$f(X) = \begin{cases} 0 & \text{if } |X| \leq 1, \\ \infty & \text{otherwise.} \end{cases}$$

Then f is additive. It is easy to see that on $G' := (V(K_2), \emptyset)$ the width function f is not additive.

Weak additivity and submodularity are preserved when passing from f to f^{mon} :

Remark 2.4.3 *Let f be a width function on a graph G .*

1. *If f is weakly submodular, then so is f^{mon} .*
2. *If f is tame, then so is f^{mon} .*

Proof. 1: Given $\varepsilon > 0$, let $X' \supseteq X$, $Y' \supseteq Y$ such that $f(X') < f^{\text{mon}}(X) + \frac{\varepsilon}{2}$ and $f(Y') < f^{\text{mon}}(Y) + \frac{\varepsilon}{2}$. Then

$$f^{\text{mon}}(X \cup Y) \leq f(X' \cup Y') \leq f(X') + f(Y') < f^{\text{mon}}(X) + f^{\text{mon}}(Y) + \varepsilon.$$

2: Obvious. □

In general, additivity is not passed on from f to f^{mon} :

Example 2.4.4 Let G be the following graph:

Graph G
vertices: 1, 2, 3
edges: {1, 2}, {2, 3}

Consider the width function f on G , where

$$f(X) = \begin{cases} 0 & \text{if } X = \emptyset, \\ 1 & \text{if } X = \{1, 2, 3\}, \\ \infty & \text{otherwise.} \end{cases}$$

It is easy to see that f is weakly submodular and additive and tame, and that f^{mon} is given by

$$f^{\text{mon}}(X) = \begin{cases} 0 & \text{if } X = \emptyset, \\ 1 & \text{otherwise.} \end{cases}$$

Since $f^{\text{mon}}(\{1\}) + f^{\text{mon}}(\{3\}) = 2 > 1 = f^{\text{mon}}(\{1, 2\})$, the function f^{mon} is not additive.

Remark 2.4.5 Let f be a width function on a graph G .

1. If f is additive, then $f(\emptyset) = 0$.
2. Suppose f is monotone. Then f is tame if and only if $f(\{v\}) \leq 1$ for every vertex $v \in V(G)$.
3. If f is monotone and weakly submodular, then $0 \leq f(X)$ for all finite subsets $X \subseteq V(G)$. If, in addition, f is tame, then

$$0 \leq f(X) \leq |X|.$$

4. Let f be monotone, tame, and weakly submodular. Then

$$0 \leq f\text{-ghw}(G) \leq \text{card-ghw}(G).$$

Proof. 1: Recall that graphs are nonempty. Choose a set $\emptyset \neq X \subseteq V(G)$. Then X and $\emptyset$ do not touch. By additivity, $f(\emptyset) + f(X) = f(X)$, and thus $f(\emptyset) = 0$.

2: Follows immediately from the definition of tameness.

3: Let $X \subseteq V(G)$ be finite. By submodularity we have

$$f(X) = f(X \cup \emptyset) \leq f(X) + f(\emptyset),$$

and thus $0 \leq f(\emptyset) \leq f(X)$ by monotonicity.

If f is also tame, then for all $v \in V(G)$ there exists a finite set $X_v \subseteq V(G)$ with $f(X_v) \leq 1$. Monotonicity implies $f(\{v\}) \leq 1$. With submodularity it follows that $f(X) \leq |X|$ for every finite set $X \subseteq V(G)$.

4: Follows from 3. □

2.4.2 Obstructions: Brambles and tangles

Up to now all invariants were defined via some sort of decompositions or strategies for the cops and they were called ‘widths’, where the ‘width’ was defined as an infimum. Now we consider obstructions to small f -cop-width, i. e. these obstructions yield winning strategies for the robber. The invariants corresponding to obstructions are called ‘numbers’ and are defined as suprema.

Definition 2.4.6 *Let f be a monotone width function on the graph G .*

- A bramble $\mathcal{B} \subseteq \mathcal{P}(V(G))$ in G is a set of subsets of $V(G)$ satisfying the following conditions:
 - (B1) Every $B \in \mathcal{B}$ is connected,
 - (B2) Any two sets $B_1, B_2 \in \mathcal{B}$ touch,
 - (B3) Every infinite subset $\mathcal{B}_0 \subseteq \mathcal{B}$ has an infinite subset $\mathcal{B}'_0 \subseteq \mathcal{B}_0$ which has non-empty intersection, i. e. $\bigcap_{B \in \mathcal{B}'_0} B \neq \emptyset$.
- The f -order of a bramble $\mathcal{B}$ is defined as

$$f\text{-order}(\mathcal{B}) = \inf\{f(Y) \in \mathbb{R} \mid Y \subseteq V(G) \text{ a finite subset, and } Y \cap B' \neq \emptyset \text{ for all } B' \in \mathcal{B}\}.$$

- The f -bramble-number of G is defined as

$$f\text{-bramble-no}(G) = \sup\{f\text{-order}(\mathcal{B}) \mid \mathcal{B} \text{ a bramble in } G\}.$$

Thus every finite clique C gives rise to a bramble $\mathcal{B} = \{\{v\} \mid v \in V(C)\}$ with $f\text{-order}(\mathcal{B}) = f(V(C))$. However, if C is an infinite clique, $\{\{v\} \mid v \in V(C)\}$ does not satisfy (B3).

In Definition 2.4.6 we could also have allowed f to be non-monotone, obtaining $f\text{-bramble-no}(G) = f^{\text{mon}}\text{-bramble-no}(G)$ for all graphs G . Note that for a finite graph G the card-bramble-number is actually the *bramble-number* of G as defined in [Re97]. Moreover, the c_H -bramble-number is actually the *hyperbramble-number* of H as defined in [AGG05].

The following result is only for completeness and will not be used.

Proposition 2.4.7 *Let f be a monotone width function on the finite graph G . Equivalent are:*

1. $f\text{-bramble-no}(G) \geq k$,
2. There is a function $\zeta : \{X \subseteq V(G) \mid f(X) \leq k\} \rightarrow \mathcal{P}(V(G))$ satisfying
 - (a) $\zeta(X)$ is a connected component of $G \setminus X$,
 - (b) If $X, Y \subseteq V(G)$ with $f(X) \leq k$ and $f(Y) \leq k$, then $\zeta(X)$ and $\zeta(Y)$ touch.

Proof. $1 \Rightarrow 2$: Let $\mathcal{B}$ be a bramble of $f\text{-order}(\mathcal{B}) \geq k$. Then for $X \subseteq V(G)$ with $f(X) \leq k$ there is an element $B_X \in \mathcal{B}$ such that $X \cap B_X = \emptyset$. Define $\zeta(X) = C$ where C is the connected component of $G \setminus X$ containing B_X , thus satisfying (a). Note that the definition is independent of the choice of B_X , because each two elements of $\mathcal{B}$ touch. The touching condition for brambles also shows that (b) is satisfied.

$2 \Rightarrow 1$: The set $\{\zeta(X) \mid X \subseteq V(G), f(X) \leq k\}$ is a bramble of f -order at least k in G . $\square$

One could call such a function ζ a *weak haven*, as it is a slightly weaker variant of a *haven* as defined in [ST93]. Weak havens were the starting point for the definition of havens for finite directed graphs in [JRST01]. The reader may wish to compare this with Proposition 2.2.11.

Proposition 2.4.8 *Let f be a monotone width function on the graph G . Then*

$$f\text{-bramble-no}(G) \leq f\text{-cw}(G).$$

Proof. Let $\mathcal{B}$ be a bramble in G of f -order at least k . We show that the robber can avoid capture by cops that only move to sets $X \subseteq V(G)$ with $f(X) < k$. The robber can win by making sure that the following invariant holds during the entire game:

In each position (X, C) (where X is the cops' position and C is the robber's escape space), C is the component of $G \setminus X$ containing a set $B_X \in \mathcal{B}$ such that $B_X \cap X = \emptyset$.

Suppose the game is in position (X, C) and the invariant holds. Note that C is unique, since any two elements of $\mathcal{B}$ touch. Now the cops move to Y with $f(Y) < k$. Then the robber moves to the component C' of $G \setminus Y$ containing a $B_Y \in \mathcal{B}$ with $B_Y \cap Y = \emptyset$. This is possible, since B_X and B_Y touch in $G \setminus (X \cup Y)$. Thus the robber can always move. Together with (B3) this shows that the robber can escape. $\square$

It is still an open question whether this inequality can be replaced by an equality.

The following invariant may seem somewhat arbitrarily defined. Yet it has the advantage of being a common lower bound for both the bramble-number and the branch-width, a quite natural invariant that will be defined in the next section.

Definition 2.4.9 *Let f be a monotone width function on the graph G .*

(B2') *A tangle in G is a bramble $\mathcal{T}$, such that $B_1, B_2, B_3 \in \mathcal{T}$ form a touching triple, i. e. $B_1 \cap B_2 \cap B_3 \neq \emptyset$, or there exists an edge $e \in E(G)$ with $e \cap B_i \neq \emptyset$, $i \in \{1, 2, 3\}$.*

The f -tangle-number of G is

$$f\text{-tangle-no}(G) = \sup\{f\text{-order}(\mathcal{T}) \mid \mathcal{T} \text{ a tangle in } G\},$$

f -order($\mathcal{T}$) being defined as for brambles.

For a finite graph G the card-tangle-number is actually the *tangle-number* of G as defined in [Re97]. The *hypertangle-number* of a hypergraph as defined in [AGG05] is not equal to the c_H -tangle-number. Since our touching conditions are more restrictive, we have less tangles, and the c_H -tangle-no($\underline{H}$) is at most the hypertangle-number of the hypergraph H . It is easy to see that the triangle K_3 satisfies $c_{K_3}\text{-tangle-no}(K_3) = 1 < 2 = c_{K_3}\text{-bramble-no}(K_3)$. Thus we cannot obtain equality here.

Theorem 2.4.10 *Let f be a monotone and weakly submodular width function on the graph G . Then*

$$f\text{-tangle-no}(G) \leq f\text{-bramble-no}(G) \leq 3 \cdot f\text{-tangle-no}(G).$$

Proof. The first inequality is true since every tangle is a bramble. For the second inequality, let $\mathcal{B}$ be a bramble in G of f -order at least k . We show that G has a tangle $\mathcal{T}$ of f -order at least $\frac{k}{3}$. Let $X \subseteq V(G)$ with $f(X) < \frac{k}{3}$. Then $V(G) \setminus X$ contains a subset $B_X \in \mathcal{B}$ with $X \cap B_X = \emptyset$. Furthermore, every $B'_X \in \mathcal{B}$ with

$X \cap B'_X = \emptyset$ lies in the same connected component C_X of $G \setminus X$ as B_X (because B_X and B'_X touch). Define

$$\mathcal{T} := \{C_X \mid X \subseteq V(G), f(X) < \frac{k}{3}\}.$$

$\mathcal{T}$ is again a bramble: (B1) and (B2) are obvious. We only show (B3): For every $C_X \in \mathcal{T}$ choose $B_X \in \mathcal{B}$ such that $B_X \subseteq C_X$. If $\{B_X \mid C_X \in \mathcal{T}\}$ is infinite, then (B3) carries over from $\mathcal{B}$ to $\mathcal{T}$. Otherwise, if $\{B_X \mid C_X \in \mathcal{T}\}$ is finite, then it contains a set B_X such that infinitely many elements of $\mathcal{T}$ are supersets of B_X . But $B_X \neq \emptyset$, since B_X touches itself. Thus (B3) holds for $\mathcal{T}$.

We now show that $\mathcal{T}$ is in fact a tangle of f -order at least $\frac{k}{3}$:

- Let $C_{X_1}, C_{X_2}, C_{X_3} \in \mathcal{T}$. Then $C_{X_1} \cap C_{X_2} \cap C_{X_3} \neq \emptyset$: By weak submodularity, $f(X_1 \cup X_2 \cup X_3) < k$. Hence there is a set $B_0 \in \mathcal{B}$ s. t. $B_0 \cap (X_1 \cup X_2 \cup X_3) = \emptyset$. Since $\mathcal{B}$ is a bramble, B_0 touches B_{X_1}, B_{X_2} and B_{X_3} . Let $C_0 \supseteq B_0$ be the connected component of $G \setminus (X_1 \cup X_2 \cup X_3)$ containing B_0 . Since C_0 touches C_{X_1} , we have $C_0 \subseteq C_{X_1}$. Similarly, $C_0 \subseteq C_{X_2}, C_{X_3}$. Hence $\emptyset \neq C_0 \subseteq C_{X_1} \cap C_{X_2} \cap C_{X_3}$.
- If $X \subseteq V(G)$ with $f(X) < \frac{k}{3}$, then $C_X \in \mathcal{T}$ and $X \cap C_X = \emptyset$. Hence the f -order of $\mathcal{T}$ is at least $\frac{k}{3}$. $\square$

2.4.3 Branch decompositions

A tree T is *subcubic*, if every vertex of T has degree at most 3.

Definition 2.4.11 Let G be a graph and let f be a monotone width function on G .

- A branch decomposition of G is a triple (T, κ, λ) as follows:
 - (BD1) T is a subcubic tree,
 - (BD2) $\kappa = (\kappa_e)_{e \in E(T)}$ associates to each tree edge $e \in E(T)$ a finite set $\kappa_e \subseteq V(G)$ of graph nodes,
 - (BD3) λ is a bijection between the set of leaves of T and $E(G)$.
 - (BD4) If a graph vertex $v \in V(G)$ is contained in two graph edges $h_1, h_2 \in E(G)$, $v \in h_1 \cap h_2$, and the path from the tree leaf $\lambda^{-1}(h_1) \in V(T)$ to the tree leaf $\lambda^{-1}(h_2) \in V(T)$ uses a tree edge $e \in E(T)$ (we say: v is separated by e), then $v \in \kappa_e$.
- The f -width of a branch decomposition is

$$f\text{-width}(T, \kappa, \lambda) = \sup\{f(\kappa_e) \mid e \in E(T)\}.$$

- The f -branch-width of G is

$$f\text{-branch-width}(G) = \inf\{f\text{-width}(T, \kappa, \lambda) \mid (T, \kappa, \lambda) \text{ is a branch decomposition of } G\}.$$

Note that that in our definition, λ maps the leaves of T to the edges of G , and not the other way round. This is slightly non-standard, but more convenient for our purposes.

A fundamental difference to the other invariants is that a graph having a branch decomposition can have at most countably many edges.

For a finite graph G , $\text{card-branch-width}(G)$ is the branch-width as defined in [RS91]. The *hyperbranch-width* of a hypergraph as defined in [AGG05] is not a special case of f -branch-width. Actually, for hypergraphs that are not the disjoint union of their hyperedges the *hyperbranch-width* of a hypergraph H is at least c_H -branch-width($\underline{H}$).

Let G be a graph with pairwise disjoint edges, and let f be a monotone width function on G . Then f -branch-width(G) = $f(\emptyset)$: We can find a branch decomposition (T, κ, λ) of G where $\kappa_e = \emptyset$ for all $e \in E(T)$. Since f is monotone, $f(\kappa_e) = f(\emptyset) = \inf\{f(X) \mid X \subseteq V(G)\}$ is optimal. This example shows that f -branch-width is fundamentally different from all f -invariants defined so far. For example if we put $f(X) := |X|$ for $|X| \leq 1$, and $f(X) := \infty$ otherwise, then the complete graph on two vertices K_2 satisfies f -branch-width(K_2) = 0 while all other invariants are infinite. Note that f is tame and monotone, though not weakly sub-modular. Since this phenomenon is the only obstruction to two inequalities that otherwise would hold, we define:

Definition 2.4.12

$$f\text{-branch-width}'(G) := \max\{f\text{-branch-width}(G), \sup_{e \in E(G)} f(e)\}.$$

In our applications $\sup_{e \in E(G)} f(e)$ will invariably be either 1 or 2, so here the difference between branch-width and branch-width' is marginal.

Let (T, κ, λ) be a branch decomposition of a graph G . For every edge $\{t, s\} \in E(T)$ we let $L_{t,s}$ and $L_{s,t}$ be the leaf sets of the two subtrees into which the tree is divided if the edge $\{t, s\}$ is removed. ($L_{t,s}$ is the leaf set of the subtree that contains s .) Let $\lambda_{t,s}$ be the set of vertices in edges contained in $\lambda(L_{t,s})$, that is,

$$\lambda_{t,s} = \bigcup \lambda(L_{t,s}) = \{v \in V(G) \mid v \in \lambda(\ell) \text{ for some leaf } \ell \in L_{t,s}\}.$$

Thus $\lambda_{s,t} \cap \lambda_{t,s} \subseteq \kappa_h$ for all edges $h = \{t, s\} \in E(T)$.

Lemma 2.4.13 *Let (T, κ, λ) be a branch decomposition of G , and let $t \in T$ be an inner tree node with neighbours $s_1, s_2, s_3 \in T$. Then the sets $\lambda_{s_i,t} \setminus \kappa_{\{s_i,t\}}$ do not form a touching triple. That is:*

1. $(\lambda_{s_1,t} \setminus \kappa_{\{s_1,t\}}) \cap (\lambda_{s_2,t} \setminus \kappa_{\{s_2,t\}}) \cap (\lambda_{s_3,t} \setminus \kappa_{\{s_3,t\}}) = \emptyset$, and
2. Every graph edge $e \in E(G)$ is disjoint from one of the sets $(\lambda_{s_i,t} \setminus \kappa_{\{s_i,t\}})$, $i = 1, 2, 3$.

Similarly if t has only two neighbours $s_1, s_2 \in T$.

Proof. First note that $\lambda_{s_1,t} = \lambda_{t,s_2} \cup \lambda_{t,s_3}$ and $\kappa_{\{s_1,t\}} \supseteq \lambda_{t,s_1} \cap \lambda_{s_1,t}$, and hence

$$\lambda_{s_1,t} \setminus \kappa_{\{s_1,t\}} \subseteq \lambda_{s_1,t} \setminus \lambda_{t,s_1} = (\lambda_{t,s_2} \cup \lambda_{t,s_3}) \setminus \lambda_{t,s_1}. \quad (2.1)$$

The analogous statement holds for every permutation of s_1, s_2, s_3 .

With this, 1 is true since

$$\begin{aligned} & (\lambda_{s_1,t} \setminus \kappa_{\{s_1,t\}}) \cap (\lambda_{s_2,t} \setminus \kappa_{\{s_2,t\}}) \cap (\lambda_{s_3,t} \setminus \kappa_{\{s_3,t\}}) \\ & \subseteq [(\lambda_{t,s_2} \cup \lambda_{t,s_3}) \setminus \lambda_{t,s_1}] \cap [(\lambda_{t,s_1} \cup \lambda_{t,s_3}) \setminus \lambda_{t,s_2}] \cap [(\lambda_{t,s_1} \cup \lambda_{t,s_2}) \setminus \lambda_{t,s_3}] \\ & = \emptyset. \end{aligned}$$

2: Let $e \in E(G)$. We may assume that $e \subseteq \lambda_{t,s_1}$. But then by (2.1)

$$\begin{aligned} \lambda_{s_1,t} \setminus \kappa_{\{s_1,t\}} & \subseteq (\lambda_{t,s_2} \cup \lambda_{t,s_3}) \setminus \lambda_{t,s_1} \\ & \subseteq (\lambda_{t,s_2} \cup \lambda_{t,s_3}) \setminus e, \end{aligned}$$

so $e \cap (\lambda_{s_1,t} \setminus \kappa_{\{s_1,t\}}) = \emptyset$. □

Proposition 2.4.14 *Let f be a monotone width function on the finite graph G . Then*

$$f\text{-tangle-no}(G) \leq f\text{-branch-width}'(G).$$

Proof. Let (T, κ, λ) be a branch decomposition of G witnessing that

$$f\text{-branch-width}'(G) \leq k.$$

For every edge $h = \{t, s\} \in E(T)$ we have $f(\kappa_h) \leq k$ and $\lambda_{t,s} \cap \lambda_{s,t} \subseteq \kappa_h$.

Suppose for contradiction that G has a tangle $\mathcal{B}$ of f -order $> k$. Then for every edge $h = \{t, s\} \in E(T)$ there is an $X_h \in \mathcal{B}$ such that $\kappa_h \cap X_h = \emptyset$. Then either $X_h \subseteq \lambda_{t,s} \setminus \kappa_h$ or $X_h \subseteq \lambda_{s,t} \setminus \kappa_h$. In the first case we let $\vec{h} = (t, s)$, in the second case we let $\vec{h} = (s, t)$. This orientation of h does not depend on the particular choice of the set $X_h \in \mathcal{B}$, because by (B2) all elements of $\mathcal{B}$ touch and hence all those that do not intersect κ_h must be on the same side. We orient all edges $h \in E(T)$ in this way. Since the resulting directed graph is finite and acyclic, there must be a node $t \in T$ with outdegree 0. First suppose t has degree 3. Let s_1, s_2, s_3 be the neighbours of t . Then $X_{s_i,t} \subseteq \lambda_{s_i,t} \setminus \kappa_{\{s_i,t\}}$ for $i = 1, 2, 3$. Since by (B2'), the sets $X_{s_1,t}, X_{s_2,t}, X_{s_3,t}$ form a touching triple we get a contradiction to Lemma 2.4.13. Similarly if t has degree 2.

Therefore t is a leaf. Let $e = \lambda(t)$, and let $\vec{f} = (s, t)$ be the edge directed towards t . Then $X_h \subseteq e$. Since by assumption $f(e) \leq k$ and since $f\text{-order}(\mathcal{B}) > k$, there must be a set $Y \in \mathcal{B}$ such that $Y \cap e = \emptyset$. Since Y and X_h touch, there is a graph edge $e' \in E(G)$ such that $e' \cap X_h \neq \emptyset$ and $e' \cap Y \neq \emptyset$. Since $e' \in \lambda(L_{t,s})$ and $e \in \lambda(L_{s,t})$, we have $e \cap e' \subseteq \kappa_h$ and hence $X_h \cap e' \subseteq \kappa_h$. Thus $X_h \cap \kappa_h \neq \emptyset$, which is a contradiction. $\square$

Proposition 2.4.15 *Let f be a monotone width function on the finite graph G . Then*

$$f\text{-branch-width}'(G) \leq f\text{-ghw}(G).$$

Proof. Let (T, B, C) be a generalised hypertree decomposition of G . We first transform this decomposition into a new decomposition (T', B', C') and define a bijection λ from the leaves of T' to $E(G)$ such that for every leaf ℓ of T' we have $B_\ell = C_\ell = \lambda(\ell)$. To achieve this, for every edge $e \in E(G)$ we pick a vertex $t_e \in T$ such that $e \subseteq B_{t_e}$. We define the tree T' by attaching a new leaf ℓ_e to t_e for every edge $e \in E(G)$. If there are other leaves in T' than the newly created leaves ℓ_e , we delete them, and if the deletion creates new leaves, we delete them as well, until the leaves ℓ_e are the only leaves of T' . For the interior vertices t of T' we let $B'_t = B_t$ and $C'_t = C_t$. For the leaves, we let $B'_{\ell_e} = C'_{\ell_e} = e$. We define the bijection λ by $\lambda(\ell_e) = e$. It is easy to see that (T', B', C') and λ have the desired properties.

In a second step, we turn T' into a subcubic tree T'' that has the same leaves as T' by repeatedly splitting nodes of degree greater than 3. For example, if t has neighbours $s_1, \dots, s_k$, where $k \geq 4$, we replace t by nodes t_1 and t_2 , connect these two nodes by an edge, and attach $s_1, \dots, s_{\lfloor k/2 \rfloor}$ to t_1 and $s_{\lfloor k/2 \rfloor + 1}, \dots, s_k$ to t_2 . We define B'' and C'' on T'' by letting the split vertices keep their pieces and guards. That is, if we split t into t_1 and t_2 , we let $B''_{t_1} = B''_{t_2} = B'_t$ and $C''_{t_1} = C''_{t_2} = C'_t$.

We obtain a generalised hypertree decomposition (T'', B'', C'') of G and a bijection λ from the leaves of T'' to $E(G)$ such that T'' is a subcubic tree, and for every leaf ℓ of T'' we have $B_\ell = C_\ell = \lambda(\ell)$.

Now let $h = \{t, s\}$ be an edge of T'' . By Lemma 2.2.15, $B_t \cap B_s$ separates the vertices in the pieces of the two parts of the tree obtained by removing the edge h . Thus in particular,

$$\kappa_h := \lambda_{s,t} \cap \lambda_{t,s} \subseteq B_t \cap B_s \subseteq C_t.$$

Since f is monotone, we have $f(\kappa_h) \leq f(C_t) \leq k$ for all edges $h \in E(T)$. Finally, for every edge $e \in E(G)$ clearly $f(e) \leq f\text{-ghw}(G)$. Thus (T'', κ, λ) is a branch decomposition of G , witnessing that $f\text{-branch-width}'(G) \leq k$. $\square$

It is not hard to see that the last result also holds for countably infinite graphs. Furthermore, it is easy to see that the triangle K_3 satisfies c_{K_3} -branch-width(K_3) = c_{K_3} -branch-width'(K_3) = $1 < 2 = c_{K_3}$ -ghw(K_3). Thus the inequality of Proposition 2.4.15 cannot be replaced by an equality. Moreover, the following inequality is tight.

Theorem 2.4.16 *Let f be a monotone, weakly submodular width function on a graph G without isolated vertices. Then*

$$f\text{-ghw}(G) \leq 2 \cdot f\text{-branch-width}'(G).$$

Proof. Let (T, κ, λ) be a branch decomposition of the graph G of f -width $k = f\text{-branch-width}'(G)$. We define a generalised hypertree decomposition (T, B, C) as follows: For an interior vertex $t \in T$, let $h_1 = \{s_1, t\}, h_2 = \{s_2, t\} \in E(T)$ be two of the edges incident with t . We let

$$\begin{aligned} B_t &= (\lambda_{t,s_1} \cap \lambda_{s_1,t}) \cup (\lambda_{t,s_2} \cap \lambda_{s_2,t}), \\ C_t &= \kappa_{h_1} \cup \kappa_{h_2}. \end{aligned}$$

For a leaf ℓ , we let $B_\ell = C_\ell = \lambda(\ell)$. Let us first argue that (T, B) is a tree-decomposition of G : For every edge $\lambda(\ell) \in E(G)$ we have $\lambda(\ell) \subseteq B_\ell$. For a vertex $v \in V(G)$, consider the set $B^{-1}(v) = \{t \in T \mid v \in B_t\}$. An interior vertex $t \in T$ belongs to this set, if at least two of the (at most three) components of $T \setminus \{t\}$ have a leaf ℓ such that $v \in \lambda(\ell)$. A leaf ℓ belongs to $B^{-1}(v)$ if $v \in \lambda(\ell)$. Thus $B^{-1}(v)$ is the union of all paths connecting leaves ℓ with $v \in \lambda(\ell)$. Clearly, this set is connected. Thus (T, B) is a tree-decomposition of G .

It follows immediately from the definition of the guards C_t that $B_t \subseteq C_t$ for all $t \in T$, thus (T, B, C) is a generalised hypertree decomposition. Since $f(\kappa_h) \leq k$ for all $h \in E(T)$, and because f is weakly submodular, we have $f(C_t) \leq f(\kappa_{h_1} \cup \kappa_{h_2}) \leq 2k$ for an inner vertex t and

$$f(C_\ell) = f(e) \leq f\text{-branch-width}'(G) = k,$$

where $\lambda(\ell) = e$, and thus $f\text{-width}(T, B, C) \leq 2k$. $\square$

2.4.4 Linking all invariants together

In this section we show that for monotone, additive, tame, weakly submodular width functions we have

$$f\text{-ghw}(G) \leq 3 \cdot \text{bramble-no}(G) + 2.$$

Together with our previous results we then have Theorem 2.4.23 below. For the proof of this inequality (and nowhere else) we need the following f -invariant.

Definition 2.4.17 *Let G be a graph, let f be a monotone width function on G .*

- *Let $M \subseteq V(G)$ be finite. A set $C \subseteq V(G)$ is M -big, if $f(M \cap C) > \frac{f(M)}{2}$.*
- *A finite set $S \subseteq V(G)$ is a balanced f -separator for the finite set $M \subseteq V(G)$, if $G \setminus S$ has no M -big connected component.*
- *Let $k \in \mathbb{R}$. A finite set $M \subseteq V(G)$ is (f, k) -linked, if every balanced f -separator S of M satisfies $f(S) > k$.*

- The f -linkedness of G is

$$f\text{-link}(G) = \sup\{k \in \mathbb{R} \mid G \text{ contains a finite } (f, k)\text{-linked set}\}.$$

Note that the card-linkedness of a finite graph G is actually the *linkedness* of G as defined in [Re97]. For example, for any clique K the vertex set $V(K)$ is $(\text{card}, \frac{|V(K)|}{2})$ -linked.

The *hyperlinkedness* of a hypergraph as defined in [AGG05] is not a special case of f -linkedness. Actually, the *hyperlinkedness* of a hypergraph H is at least $c_H\text{-link}(\underline{H})$.

The f -linkedness is defined as an obstruction number, but it can also be viewed as a decomposition width:

Remark 2.4.18 Let f be a monotone width function on the graph G . Then

$$f\text{-link}(G) = \inf\{k \in \mathbb{R} \mid \text{each finite } M \subseteq V(G) \text{ has a balanced } f\text{-separator } S_M \text{ with } f(S_M) \leq k\}.$$

Proof. $f\text{-link}(G) = \sup\{k \in \mathbb{R} \mid G \text{ contains a finite } (f, k)\text{-linked set}\}$
 $= \inf\{k \in \mathbb{R} \mid \text{each finite } M \subseteq V(G) \text{ has a balanced } f\text{-separator } S_M \text{ with } f(S_M) \leq k\}.$ □

Lemma 2.4.19 Let f be a monotone, additive width function on the graph G . Let $M \subseteq V(G)$ be finite. Then any two M -big sets $C_1, C_2 \subseteq V(G)$ touch. In particular, for any $S \subseteq V(G)$ there is at most one M -big component of $G \setminus S$.

Proof. Towards a contradiction, suppose that C_1 and C_2 do not touch. Then $M \cap C_1$ and $M \cap C_2$ do not touch, and additivity implies that

$$\begin{aligned} \frac{f(M)}{2} + \frac{f(M)}{2} &< f(M \cap C_1) + f(M \cap C_2) \\ &\leq f((M \cap C_1) \cup (M \cap C_2)) \\ &\leq f(M), \end{aligned}$$

where the last inequality holds by monotonicity, and we have the desired contradiction. □

Proposition 2.4.20 Let f be a monotone, additive width function on the graph G . Then

$$f\text{-link}(G) \leq f\text{-bramble-no}(G).$$

Proof. Every (f, k) -linked set M generates a bramble of f -order at least k : Define

$$\mathcal{B} := \{C \mid C \text{ the } M\text{-big component of } G \setminus S \text{ for some finite } S \subseteq E(H) \text{ with } f(S) \leq k\}.$$

By Lemma 2.4.19 any two elements of $\mathcal{B}$ touch. Moreover, no set $S \subseteq V(G)$ with $f(S) < k$ intersects every element of $\mathcal{B}$. Hence $\mathcal{B}$ is a bramble of order at least k . □

In [AGG05] it is shown that K_5 has hyperlinkedness at most two and hyperbramble-number at least three. Since $c_H\text{-bramble-no}(\underline{H})$ equals the hyperbramble-number of H it follows that $c_{K_5}\text{-link}(K_5) \leq 2 < 3 \leq c_{K_5}\text{-bramble-no}(K_5)$. Thus we cannot obtain equality here.

The following technical lemma will only be used in the proof of Theorem 2.4.22.

Lemma 2.4.21 *Let G be a graph and let f be a monotone, weakly submodular width function on G . Suppose that $f\text{-link}(G) \leq k$ for some $k \in \mathbb{R}$. Let $M \subseteq V(G)$ with $f(M) \leq 2k + 2$. Then there exists a set $X' \subseteq V(G)$ with $M \subseteq X'$ and $f(X') \leq 3k + 2$ such that for all components R of $G \setminus X'$ there is a subset $M' \subseteq X'$ with $f(M') \leq 2k + 1$ and $\partial R' \subseteq M'$.*

Where for $X \subseteq V(G)$ and a connected component R of $G \setminus X$,

$$\partial R = \{v \in X \mid \text{there is an edge } e \in E(G) \text{ with } v \in e \text{ and } e \cap R \neq \emptyset\}.$$

Proof. Given M , let S be a balanced f -separator for M with $f(S) \leq k$. Set $X' := M \cup S$. Then $f(X') = f(M \cup S) \leq f(M) + f(S) \leq 2k + 2 + k$ because f is weakly submodular. Let R' be a component of $G \setminus X'$. Then $R' \subseteq C$ for a component C of $G \setminus S$. Since S is a balanced f -separator for M , C is not M -big, i. e.

$$f(C \cap M) \leq \frac{f(M)}{2} \leq \frac{2k + 2}{2} = k + 1.$$

Set $M' := S \cup (C \cap M)$. Then $M' \subseteq X'$ and with weak submodularity

$$f(M') \leq f(S) + f(C \cap M) \leq k + (k + 1) = 2k + 1.$$

Furthermore, $\partial R' \subseteq S \cup (C \cap M) = M'$. □

Theorem 2.4.22 *Let G be a finite or countably infinite graph, $k \in \mathbb{R}$, and let f be a monotone, additive, tame, weakly submodular width function on G . Then*

$$f\text{-ghw}(G) \leq 3 \cdot (f\text{-link}(G)) + 2.$$

Proof. Let f satisfy the required conditions. Let $f\text{-link}(G) \leq k$. Since f is monotone, by Remark 2.1.18 and Theorem 2.2.12, $f\text{-ghw}(G) = f\text{-hw}(G) = f\text{-cw}_{\text{mon}}(G)$ and it suffices to show that the cops have a monotone winning strategy of f -width at most $3k + 2$: The cops can make sure that each position (X, R) satisfies $f(X) \leq 3k + 2$ and $\partial R \subseteq M$ for an $M \subseteq X$ with $f(M) \leq 2k + 1$:

Suppose this is true for (X, R) . Choose $v \in R$. Since f is monotone, tameness implies that $f(\{v\}) \leq 1$. By weak submodularity we have $f(M \cup \{v\}) \leq 2k + 2$. Application of Lemma 2.4.21 to $M \cup \{v\}$ yields a set $X' \supseteq M \cup \{v\}$ with $f(X') \leq 3k + 2$, s. t. for each possible escape space R' with respect to X' there exists an $M' \subseteq X'$ with $f(M') \leq 2k$ and $\partial R' \subseteq M'$.

Obviously, this strategy is monotone, and in the case that G is finite, it is a winning strategy. If G is countably infinite, then in each step the cops have to choose the vertex v in such a way, that during the game each vertex of $V(G)$ is chosen. □

Putting things together we get:

Theorem 2.4.23 *Let f be a monotone width function on a finite graph G .*

1. *We have $f\text{-ghw}(G) = f\text{-hw}(G) = f\text{-cw}_{\text{mon}}(G)$ and the following inequalities:*

$$f\text{-tangle-no}(G) \leq f\text{-bramble-no}(G) \leq f\text{-cw}(G) \leq f\text{-ghw}(G)$$

$$f\text{-tangle-no}(G) \leq f\text{-branch-width}'(G) \leq f\text{-ghw}(G)$$

2. *If f is also weakly submodular, then we also have*

$$f\text{-bramble-no}(G) \leq 3 \cdot f\text{-tangle-no}(G),$$

(and

$$f\text{-ghw}(G) \leq 2 \cdot f\text{-branch-width}'(G),$$

provided that G has no isolated vertices).

3. If f is also weakly submodular, additive and tame, then

$$f\text{-ghw}(G) \leq 3 \cdot f\text{-bramble-no}(G) + 2.$$

Thus, all invariants occurring in the theorem are linearly coherent on the class of graphs equipped with monotone, weakly submodular, additive, tame width functions. (Here we have omitted f -linkedness, because it has a rather artificial definition.)

Chapter 3

Hypergraphs

In this Chapter we study the width function c_H , with main focus on the relationship between c_H -cw, c_H -ghw, and c_H -hw.¹

In Section 3.1.1 we introduce *hypergraph pairs*, i. e. pairs (G, H) of hypergraphs with $V(G) = V(H)$. Obviously, the function c_H is a width function on $\underline{G}$. Then the properties of c_H are listed which follow from the general theory in Chapter 2.

Hypergraph pairs will be helpful in several places for constructing examples of hypergraphs. As mentioned before, in [AGG05] it was shown that any finite hypergraph H satisfies

$$c_H\text{-cw}(H) \leq k \implies c_H\text{-hw}(H) \leq 3k + 1.$$

Thus, since $c_H\text{-cw}(H) \leq c_H\text{-ghw}(H) \leq c_H\text{-hw}(H)$, all three invariants are linearly coherent.

In Section 3.2 we first construct hypergraph pairs (G_n, H_n) satisfying

$$c_{H_n}\text{-cw}(G_n) = n \text{ and } c_{H_n}\text{-ghw}(G_n) = 2n$$

for every integer $n > 0$. Thus for hypergraph pairs (G, H) the invariants $c_H\text{-cw}(G)$ and $c_H\text{-ghw}(G)$ are not strongly coherent. This result actually carries over to hypergraphs. This is done by ‘implementing’ a hypergraph pair (G, H) with $\underline{H} \subseteq \underline{G}$ as a hypergraph H' in such a way that $c_H\text{-cw}(G) = c_{H'}\text{-cw}(H')$ and $c_H\text{-ghw}(G) = c_{H'}\text{-ghw}(H')$. Since the hypergraph pairs (G_n, H_n) constructed in Section 3.2 satisfy $\underline{H_n} \subseteq \underline{G_n}$, this shows that even for hypergraphs H the invariants $c_H\text{-cw}(H)$ and $c_H\text{-ghw}(H)$ are not strongly coherent. Apart from this, the implementation method will be useful in Chapter 5.

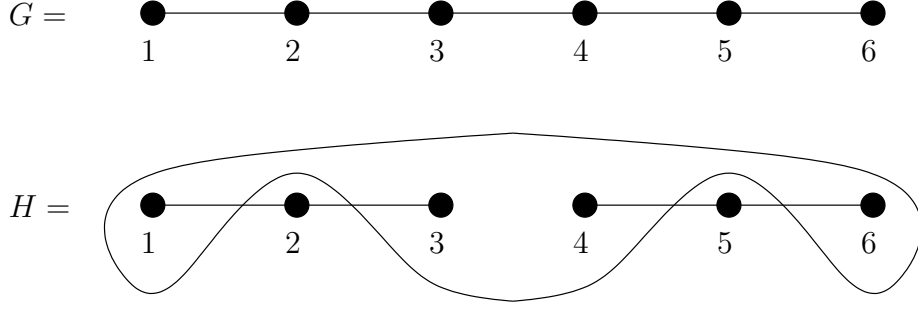
In Section 3.3, the relationship between the invariants $c_H\text{-ghw}$ and $c_H\text{-hw}$ is discussed. Since in [Ad02] (see also [Ad04]) it was already shown that they are not strongly coherent on hypergraphs, we only give an easy example of a hypergraph H with $c_H\text{-ghw}(H) = 2 < 3 = c_H\text{-hw}(H)$. Then we show that for every integer $n > 0$ there exists a hypergraph pair (G_n, H_n) satisfying

$$c_{H_n}\text{-ghw}(G_n) = 1 \text{ and } c_{H_n}\text{-hw}(G_n) = n.$$

Thus, $c_H\text{-ghw}$ and $c_H\text{-hw}$ are not coherent on hypergraph pairs.

Finally, in Section 3.4 we show that deciding for fixed $k < \omega$ whether a given finite hypergraph pair (G, H) satisfies $c_H\text{-ghw}(G) \leq k$ is actually equivalent to solving a well known problem: The *Hypergraph Sandwich Problem* defined by A. Lustig, O. Shmueli in [LS99]. It is an open question, whether the hypergraph sandwich problem is solvable in polynomial time. We show that restriction to inputs with bounded maximum hyperedge size leads to a polynomial time algorithm.

¹Parts of this chapter are based on [Ad05a].

Figure 3.1: A hypergraph pair (G, H) .

3.1 Definitions and some observations

3.1.1 Hypergraph pairs

Definition 3.1.1

1. A hypergraph pair is a pair (G, H) , where G and H are hypergraphs and $V(G) = V(H)$.
2. A hypergraph pair (G_0, H_0) is an induced subhypergraph pair of (G, H) , if G_0 is an induced subhypergraph of G and H_0 is an induced subhypergraph of H .

Figure 3.1 shows a hypergraph pair.² The only thing we will study for hypergraph pairs is c_H -invariants (more precisely: c_H -cw, c_H -ghw, and c_H -hw) applied to $\underline{G}$. Therefore, whenever we need it, we may assume that a hypergraph pair (G, H) satisfies $G = \underline{G}$. Intuitively, G measures the connectivity ('robber graph') and H measures the cost of covering a set of vertices of G ('cop graph').

Example 3.1.2 The following hypergraph pair is depicted in Figure 3.1.

Hypergraph pair (G, H)	
vertices:	1, 2, 3, 4, 5, 6
edges of G :	$\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{5, 6\}$
edges of H :	$\{1, 2\}, \{2, 3\}, \{1, 3, 4, 6\}, \{4, 5\}, \{5, 6\}$

Figure 3.2 shows a hypertree decomposition of the graph G from Example 3.1.2, of c_H -width 2. Figure 3.3 shows a generalised hypertree decomposition of G , of c_H -width 1. Each tree node t is depicted with B_t on the left hand side and C_t on the right hand side. Actually, these decompositions are optimal in the sense that $c_H\text{-hw}(G) = 2$ and $c_H\text{-ghw}(G) = 1$. The proof is left to the reader. In Section 3.3 we will 'implement' examples such as this in order to obtain hypergraphs H satisfying $c_H\text{-hw}(H) > c_H\text{-ghw}(H)$.

²Hypergraph pairs are in fact quite natural. For example, in [FFG02] J. Flum, M. Frick and M. Grohe define the notion of acyclicity for a conjunctive query with negation, i. e. for a formula φ which is a conjunction of relations and negated relations. Let $\text{at}^+(\varphi)$ (and $\text{at}^-(\varphi)$, resp.) denote the set of all relations occurring positively (negatively, resp.) in φ . There is a natural hypergraph pair (G_φ, H_φ) corresponding to φ :

Hypergraph pair (G_φ, H_φ)	
vertex set:	$\text{var } \varphi$
edges of G_φ :	$h \subseteq \text{var}(\varphi), h \in \text{at}^+(\varphi) \cup \text{at}^-(\varphi)$
edges of H_φ :	$h \subseteq \text{var}(\varphi), h \in \text{at}^+(\varphi)$

It is straightforward to see that φ is *acyclic* in the sense of [FFG02] if and only if the hypergraph pair (G_φ, H_φ) satisfies $c_{H_\varphi}\text{-hw}(G_\varphi) \leq 1$.

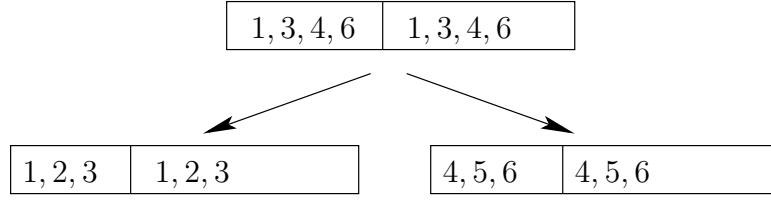


Figure 3.2: A hypertree decomposition of G from Figure 3.1 of c_H -width 2. Every node $t \in T$ is depicted by a box with B_t on the left hand side and C_t on the right hand side.

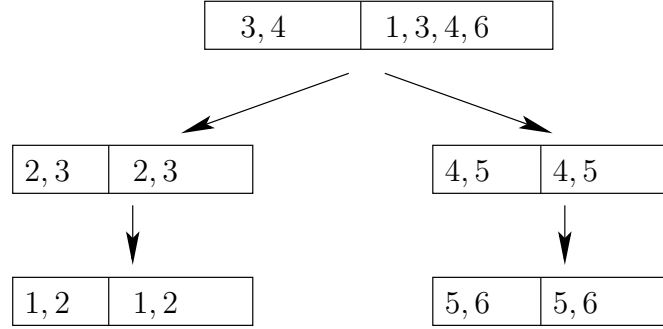


Figure 3.3: A generalised hypertree decomposition of G from Figure 3.1 of c_H -width 1.

We obtain the following special case:

Remark 3.1.3

Let (G, H) be a hypergraph pair with $E(H) = \{\{v\} \mid v \in V(G)\}$. Then

$$\text{tw}(G) + 1 = c_H - \text{ghw}(G) = c_H - \text{hw}(G) = c_H - \text{cw}(G).$$

Proof. The first two equalities follow from the definitions, and the last equality holds since by Seymour and Thomas [ST93] there is no monotonicity cost on graphs.

Example 3.1.4 Let (G, H) be a hypergraph pair. Then

- $c_H - \text{hw}(G)$ is the hypertree-width of (G, H) as defined in [Ad05a], and
- $c_H - \text{ghw}(G)$ is the generalised hypertree-width of (G, H) as defined in [Ad05a].

The games defined in [GLS01b] and [Ad05a] can be regarded as special cases³ of $\text{RC}(G, f, k)$, and we get:

Example 3.1.5

- Let H be a finite hypergraph. Then $c_H - \text{cw}(H)$ (or $c_H - \text{cw}_{\text{mon}}(H)$) is the minimum number of marshals necessary to catch the robber in the (monotone, respectively) robber and marshals game defined in [GLS01b].
- Let (G, H) be a hypergraph pair. Then $c_H - \text{cw}(G)$ (or $c_H - \text{cw}_{\text{mon}}(G)$) is the minimum number of marshals necessary to catch the robber in the (monotone, respectively) robber and marshals game defined in [Ad05a].

³Recall that the robber and cops game defined by Seymour and Thomas in [ST93] can also be regarded as a special case (cf. Example 2.2.9).

3.1.2 Properties of c_H

In order to apply Chapter 2 to $f = c_H$, we first take a look at the properties of c_H (cf. Definition 2.4.2).

Lemma 3.1.6

1. For a hypergraph pair (G, H) , the width function c_H^{mon} on G is weakly submodular.
2. For a hypergraph H , the width function c_H^{mon} on H is weakly submodular and additive. If H is tame, then so is c_H^{mon} .

Proof. Recall that by Remark 2.4.3, weak submodularity and tameness are passed on from f to f^{mon} . We use this freely.

1: Weak submodularity of c_H : Let (G, H) be a hypergraph pair and let X_1, X_2 be finite subsets of $V(G)$. If $c_H(X_1) + c_H(X_2) = \infty$, then there is nothing to show. Otherwise, there exist $E_1, E_2 \subseteq E(H)$ such that $\bigcup E_i = X_i$ and $|E_i| = c_H(X_i)$ for $i \in \{1, 2\}$. Then $\bigcup(E_1 \cup E_2) = X_1 \cup X_2$ and thus $c_H(X_1 \cup X_2) \leq c_H(X_1) + c_H(X_2)$.

2: Weak submodularity of c_H^{mon} holds by 1.

Additivity of c_H^{mon} : Since c_H^{mon} is weakly submodular, we only need to show that

$$c_H^{\text{mon}}(X \cup Y) \geq c_H^{\text{mon}}(X) + c_H^{\text{mon}}(Y)$$

for finite non-touching sets $X, Y \subseteq V(H)$. If $c_H^{\text{mon}}(X \cup Y) = \infty$, then there is nothing to show. Otherwise, let $E_{X \cup Y} \subseteq E(H) \setminus \{\emptyset\}$ be such that

$$\bigcup E_{X \cup Y} \supseteq X \cup Y \text{ and } |E_{X \cup Y}| = c_H^{\text{mon}}(X \cup Y).$$

Let $E_X = \{h \in E_{X \cup Y} \mid h \cap X \neq \emptyset\}$ and $E_Y = \{h \in E_{X \cup Y} \mid h \cap Y \neq \emptyset\}$. Since X and Y do not touch, $E_{X \cup Y}$ is the disjoint union $E_{X \cup Y} = E_X \dot{\cup} E_Y$. Since $\bigcup E_X \supseteq X$ and $\bigcup E_Y \supseteq Y$, we have

$$c_H^{\text{mon}}(X) + c_H^{\text{mon}}(Y) \leq |E_X| + |E_Y| = |E_{X \cup Y}| = c_H^{\text{mon}}(X \cup Y).$$

Tameness of c_H : If H has no isolated vertices, then every vertex v is contained in a hyperedge h . Now clearly $c_H(h) = 1$. $\square$

Corollary 3.1.7

- For a hypergraph pair (G, H) , the width function c_H^{mon} on G satisfies Theorem 2.4.23,1 and 2.
- For a hypergraph H , the width function c_H^{mon} on H satisfies Theorem 2.4.23,1 and 2. If H is tame, then c_H^{mon} also satisfies Theorem 2.4.23,3.

In general, additivity does not hold for the width function c_H of a hypergraph pair (G, H) :

Example 3.1.8 Consider the following hypergraph pair.

Hypergraph pair (G, H)	
vertices:	1, 2
edges of G :	$\{1\}, \{2\}$
edges of H :	$\{1\}, \{2\}, \{1, 2\}$

The hyperedges $\{1\}$ and $\{2\}$ do not touch in G , but

$$c_H(\{1, 2\}) = 1 < 1 + 1 = c_H(\{1\}) + c_H(\{2\}).$$

The following theorem shows that even c_H -ghw and c_H -hw are linearly coherent.

Fact 3.1.9 ([AGG05]) *Every finite hypergraph H satisfies*

$$c_H\text{-hw}(H) \leq 3 \cdot c_H\text{-ghw}(H) + 1.$$

Lemma 3.1.10 *Let f be the family of width functions c_H , for all hypergraphs H . For $k > 0$, define $\mathcal{C}_k^f := \{\text{structures } \mathcal{M} \mid c_{H_{\mathcal{M}}}(H_{\mathcal{M}}) \leq k\}$. Then*

- $\text{SUBSTR}_{\text{enum}}^{\mathcal{C}_k^f} \in \text{P}$, and
- $\text{HOM}_{\text{enum}}(\mathcal{C}_k^f, -) \in \text{P}$.

Corollary 3.1.11 *Let f be the family of width functions c_H , for all hypergraphs H , and let $k > 0$ be an integer. Then*

1. $\text{HOM}(\text{COREDECOMPOSABLE}_{\text{ghd}}^{\mathcal{C}_k^f}, -) \in \text{P}$,
2. $\text{HD}^{\mathcal{C}_k^f} \in \text{P}$,
3. $\text{HOM}(\text{DECOMPOSED}^{\mathcal{C}_k^f}, -) \in \text{P}$,
4. $\text{CS}^{\mathcal{C}_k^f} \in \text{P}$.

Proof. Use Theorems 2.3.13, 2.3.16, 2.3.18, and Corollary 2.3.19. $\square$

Corollary 3.1.11, 2 and 3 actually reprove results from G. Gottlob, N. Leone and F. Scarcello in [GLS02].

3.1.3 c_H -hypertree-width is smaller than tree-width

The hypergraph invariants $c_H\text{-hw}(H)$ and $\text{card-hw}(H)$ ($= \text{tw}(H) + 1$) are not coherent:

Example 3.1.12 *For $n < \omega$ consider the following hypergraph.*

Hypergraph H
vertices: $1, 2, \dots, n$
edges: $\{1, 2, 3, \dots, n\}$

Clearly $c_H\text{-hw}(H) = 1$ and $\text{card-hw}(H) = n$.

However, when we pass from H to the underlying graph $\underline{H}$ we loose some information. We might get better results by actually encoding H in a graph and then taking the tree-width. Therefore we also consider the following definition.

Definition 3.1.13 *For a hypergraph H , the incidence graph $H^\times$ is the bipartite graph defined as follows.*

Graph $H^\times$
vertex set: $V(H) \dot{\cup} E(H)$
edges: $\{v, e\} \in V(H) \times E(H), \text{ where } v \in e$

But again we have no luck:

Example 3.1.14 *For $n < \omega$ consider the following hypergraph.*

Hypergraph H	
vertices:	$1, 2, \dots, n$
edges:	$\{1, 2, 3, \dots, n\}$ $\{i, j\}, \text{ where } 1 \leq i < j \leq n$

Clearly $c_H\text{-hw}(H) = 1$ and $\text{card-hw}(H^\times) = n + 1$.

On the other hand we have the following inequalities.

Proposition 3.1.15 *Let H be a tame hypergraph. Then*

1. $c_H\text{-hw}(H) \leq \text{card-hw}(H)$, and
2. $c_H\text{-hw}(H) \leq \text{card-hw}(H^\times)$.

Proof. 1: Let (T, B, B) be a hypertree decomposition of H with

$$\text{card-width}(T, B, B) \leq k.$$

Let $\varphi : V(H) \rightarrow E(H)$ be an arbitrary function such that $v \in \varphi(v)$. Define $C_t := \bigcup \{\varphi(v) \mid v \in B_t\}$. We claim that (T, C, C) is a hypertree decomposition of H such that $c_H\text{-width}(T, C, C) \leq \text{card-width}(T, B, B)$. We only prove the connectedness condition (TD3), as this is slightly tricky. Given $v \in V(H)$ we have

$$\begin{aligned} \{t \in T \mid v \in C_t\} &= \{t \in T \mid \text{there is a vertex } w \in B_t \text{ such that } v \in \varphi(w)\} \\ &= \bigcup_{\substack{w \in V(H) \\ v \in \varphi(w)}} \{t \in T \mid w \in B_t\}. \end{aligned}$$

Of course, the set $\{w \in V(H) \mid v \in \varphi(w)\} \subseteq V(H)$ is connected. Thus by Remark 1.1.3, the set

$$\bigcup_{\substack{w \in V(H) \\ v \in \varphi(w)}} \{t \in T \mid w \in B_t\} = \{t \in T \mid v \in C_t\}$$

is connected in T .

2: Let (T, B, B) be a hypertree decomposition of $H^\times$ with

$$\text{card-width}(T, B, B) \leq k.$$

Let $\varphi : V(H) \dot{\cup} E(H) \rightarrow E(H)$ be an arbitrary function such that $v \in \varphi(v)$ and $e = \varphi(e)$. Let $C_t = \bigcup \{\varphi(b) \mid b \in B_t\}$ as before. Again we claim that (T, C, C) is a hypertree decomposition of H such that $c_H\text{-width}(T, C, C) \leq \text{card-width}(T, B, B)$. For $v \in V(H)$,

$$\begin{aligned} \{t \in T \mid v \in C_t\} &= \{t \in T \mid \text{there is a vertex } w \in B_t \cap V(H) \text{ such that } v \in \varphi(w)\} \\ &\quad \cup \{t \in T \mid \text{there is a vertex } w \in B_t \cap E(H) \text{ such that } v \in \varphi(w)\} \\ &= \bigcup_{\substack{w \in V(H) \\ v \in \varphi(w)}} \{t \in T \mid w \in B_t\} \cup \bigcup_{\substack{e \in E(H) \\ v \in e}} \{t \in T \mid e \in B_t\}. \end{aligned}$$

Each of these sets is connected. Let $e \in E(H)$ be such that $v \in e$. Then the sets $\{t \in T \mid e \in B_t\}$ and $\{t \in T \mid v \in B_t\}$ intersect because there must be a tree node $s \in T$ such that $\{e, v\} \subseteq B_s$. Similarly, if $w \in V(H)$ is a graph vertex such that $v \in \varphi(w)$, then $\{t \in T \mid \varphi(w) \in B_t\}$ and $\{t \in T \mid w \in B_t\}$ intersect. $\square$

If we have an upper bound on the size of the hyperedges, then the invariants are linearly coherent:

Definition 3.1.16 A hypergraph H is bounded, if there is an integer $\kappa \geq 0$ such that all $e \in E(H)$ satisfy $|e| \leq \kappa$.

Remark 3.1.17 Let H be a hypergraph bounded by κ . Then

1. $\text{card-hw}(H) \leq \kappa \cdot c_H\text{-hw}(H)$, and
2. $\text{card-hw}(H^\times) \leq \kappa \cdot c_H\text{-hw}(H) + 1$.

Proof. The proof is left to the reader. Hint: Use the game theoretic characterisation and the fact that κ cops can cover any hyperedge of H . $\square$

3.2 The relation of c_H -cw and c_H -ghw

By Theorem 2.4.23, 1 and 3, applied to $f = c_H^{\text{mon}}$, it follows that c_H -cw and c_H -ghw are linearly coherent for finite hypergraphs H . With this, in this section we show:

- For hypergraphs H , c_H -cw and c_H -ghw are linearly coherent, but not strongly coherent.
- Then same is true for hypergraph pairs (G, H) satisfying $\underline{H} \subseteq \underline{G}$.

It remains an open question, whether this is true for arbitrary hypergraph pairs.

3.2.1 Hypergraph pairs

In [Ad04] a graph G is presented with $c_G\text{-cw}(G) = 4$ and $c_G\text{-ghw}(G) = 5$. Are there hypergraph pairs or even hypergraphs where these invariants differ by more than one? In this section we will show that the answer to both questions is yes. We begin with a hypergraph pair (G, H) with $c_H\text{-cw}(G) = 1$ and $c_H\text{-ghw}(G) = 2$. Then we show how we can ‘multiply’ a hypergraph pair (G, H) , to obtain a hypergraph pair $(G \times n, H \cdot n)$ with $c_{H \cdot n}\text{-cw}(G \times n) = n \cdot c_H\text{-cw}(G)$ and $c_{H \cdot n}\text{-ghw}(G \times n) = n \cdot c_H\text{-ghw}(G)$.

Definition 3.2.1 Let H be a hypergraph. The simplicialisation of H is the following hypergraph ΣH .

Hypergraph ΣH	
vertex set:	$V(H)$
edges:	e' , where $e' \subseteq e$ for an edge $e \in E(H)$

The following Remark holds due to $c_H^{\text{mon}} = c_{\Sigma H}$. It was proved in [Ad02] (see also [Ad04]).

Remark 3.2.2 A hypergraph pair (G, H) satisfies $c_H\text{-ghw}(G) = c_{\Sigma H}\text{-hw}(G)$.

Consider the following hypergraph pair (G, H) (see Figure 3.4):

Hypergraph pair (G, H)	
vertices:	$A, 1, 2, 3, 4, B$
edges of G :	$\{A, 1\}, \{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, B\}, \{A, B\}$
edges of H :	$\{A, 1, B\}, \{1, 2, B\}, \{2, 3, A, B\}, \{3, 4, A\}, \{4, A, B\}, \{A, B\}$

Remark 3.2.3 The hypergraph pair (G, H) defined above satisfies

$$c_H\text{-cw}(G) = 1 \text{ and } c_H\text{-ghw}(G) = 2.$$

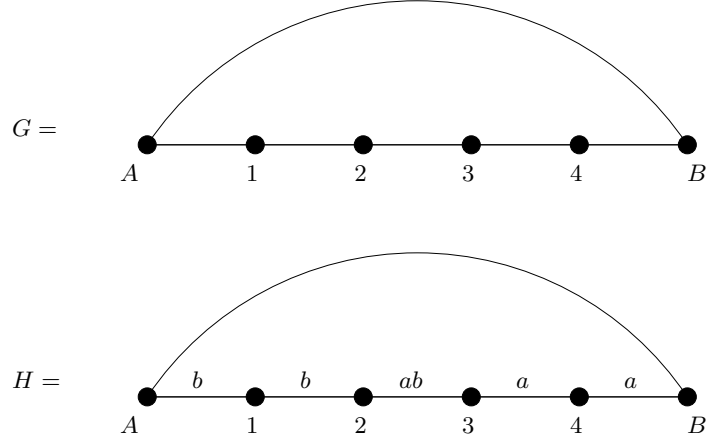


Figure 3.4: The hypergraph pair (G, H) . The hyperedges of H marked by a (resp. b) additionally contain the vertex A (resp. B).

Proof. One cop can catch the robber chasing him from left to right. Note that the move from $\{2, 3, A, B\}$ to $\{3, 4, A\}$ is non-monotone.

Show $c_H\text{-ghw}(G) = 2$ by showing that $c_{\Sigma H}\text{-hw}(G) = 2$ (cf. Remark 3.2.2): It is easy to see that two cops can win monotonely on $(G, \Sigma H)$. It is not hard to see that the robber can escape from one monotone cop: consider all possible starting positions of the cop. $\square$

Definition 3.2.4

- For a graph G we define the graph $G \times n$ as follows.

Graph $G \times n$	
vertices:	$(v, 1), \dots, (v, n)$, where $v \in V(G)$
edges:	$\{(v, i), (w, j)\}$, where $\{v, w\} \in E(G)$ and $1 \leq i \leq j \leq n$
	$\{(v, i), (v, j)\}$, where $v \in V(G)$ and $1 \leq i < j \leq n$

If G is a hypergraph, then $G \times n$ denotes $\underline{G} \times n$.

- For a hypergraph H we define the hypergraph $H \cdot n$ as follows.

Hypergraph $H \cdot n$	
vertices:	$(v, 1), \dots, (v, n)$, where $v \in V(H)$
edges:	$h \times \{1\}, \dots, h \times \{n\}$ where $h \in E(H)$

Note that $H \cdot n$ is the hypergraph that consists of n disjoint copies of H . $G \times n$ contains n copies of G which are pairwise connected by new edges. Figure 3.5 shows the hypergraph pair $(G \times 2, H \cdot 2)$ for (G, H) of Figure 3.4.

For a hypergraph pair (G, H) recall that in the game $\text{RC}(G, c_H, k)$, the cops may only occupy subsets $X \subseteq V(G)$ satisfying $c_H(X) \leq k$. Equivalently, we can say that the cops may only occupy sets $Y \subseteq E(H)$ of at most k hyperedges (then $c_H(\bigcup Y) \leq k$). We will use this equivalence freely throughout this chapter.

Theorem 3.2.5 *Let (G, H) be a finite hypergraph pair. Then:*

1. $c_{(H \cdot n)}\text{-cw}(G \times n) = n \cdot (c_H\text{-cw}(G))$.
2. $c_{(H \cdot n)}\text{-ghw}(G \times n) = n \cdot (c_H\text{-ghw}(G))$.

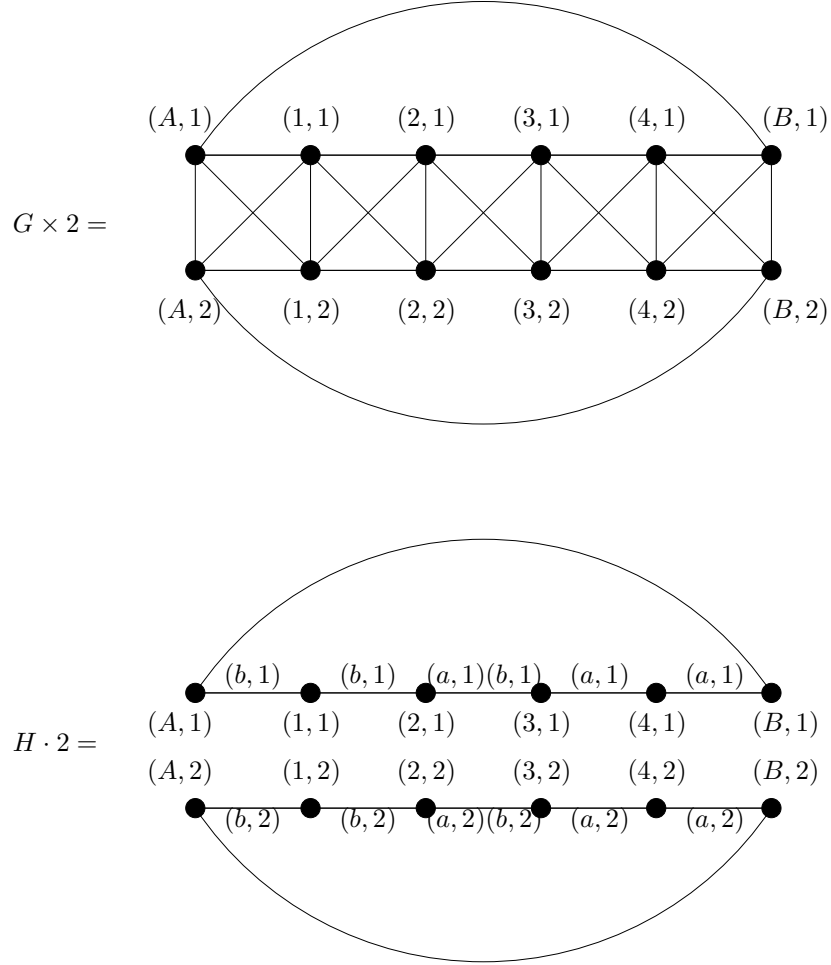


Figure 3.5: The hypergraph pair $(G \times 2, H \cdot 2)$ for (G, H) of Figure 3.4. The hyperedges of $H \cdot 2$ made by (a, i) (resp. (b, i)) additionally contain the vertex (A, i) (resp. (B, i)), for $i=1,2$.

Proof. Since for any hypergraph pair $c_{H\cdot n}\text{-cw}(G) = c_{\Sigma H}\text{-cw}(G)$ and $c_{H\cdot n}\text{-ghw}(G) = c_{\Sigma H}\text{-cw}_{\text{mon}}(G)$, during the proof we assume that H , and thus $H \cdot n$, are simplicial, i. e. $H = \Sigma H$ and $H \cdot n = \Sigma(H \cdot n)$.

Given a winning strategy (a monotone winning strategy) on (G, H) for k cops, it is not hard to see that we obtain a winning strategy (a monotone winning strategy) on $(G \times n, H \cdot n)$, by moving simultaneously on each of the n copies of (G, H) in $(G \times n, H \cdot n)$ as on (G, H) . Thus $c_{(H \cdot n)}\text{-cw}(G \times n) \leq n \cdot (c_H\text{-cw}(G))$ and $c_{(H \cdot n)}\text{-ghw}(G \times n) \leq n \cdot (c_H\text{-ghw}(G))$.

For the converse, we define maps

- $\pi : V(G \times n) \rightarrow V(G)$, $\pi((v, i)) := v$,
- $\pi^+ : \mathcal{P}(V(G \times n)) \rightarrow \mathcal{P}(V(G))$, $\pi^+(R) := \{\pi(v) \mid v \in R\}$,
- $\pi^- : \mathcal{P}(V(G \times n)) \rightarrow \mathcal{P}(V(G))$, $\pi^-(X) := \{v \in V(G) \mid \pi^{-1}(v) \subseteq X\}$
 $= V(G) \setminus \pi^+(V(G \times n) \setminus X)$.

Suppose k cops have a winning strategy (a monotone winning strategy) on the hypergraph pair $(G \times n, H \cdot n)$. We obtain a winning strategy for $\lfloor \frac{k}{n} \rfloor$ cops on (G, H) as follows: Let $X_1 \subseteq E(H \cdot n)$ be the first position of the cops in their winning strategy on $(G \times n, H \cdot n)$. For the first move on (G, H) , $\lfloor \frac{k}{n} \rfloor$ cops choose a set $X'_1 \subseteq E(H)$ with $\bigcup X'_1 = \pi^-(\bigcup X_1)$. This is possible since H is simplicial and since there is one copy of H in $H \cdot n$ containing at most $\lfloor \frac{k}{n} \rfloor$ edges of X_1 and these edges must cover all copies of elements of $\pi^-(\bigcup X_1)$.

Now the robber chooses an escape space $R'_1 \subseteq V(G)$ with respect to X'_1 . It is easy to see that there is an escape space $R_1 \subseteq V(G \times n)$ w. r. t. X_1 such that $\pi^+(R_1) = R'_1$. For the case of the robber choosing R_1 there is an answer $X_2 \subseteq E(H \cdot n)$ for the k cops according to their winning strategy on $(G \times n, H \cdot n)$. On (G, H) , $\lfloor \frac{k}{n} \rfloor$ cops again choose a set $X'_2 \subseteq E(H)$ with $\bigcup X'_2 = \pi^-(\bigcup X_2)$. Now it is the robber's turn to choose a possible escape space $R'_2 \subseteq V(G)$. As above, there is an escape space $R_2 \subseteq V(G \times n)$ w. r. t. X_2 such that $\pi^+(R_2) = R'_2$. It is not hard to see that since R'_1 and R'_2 are connected in $G \setminus (\bigcup X'_1 \cap \bigcup X'_2)$, R_1 and R_2 are connected in $(G \times n) \setminus (\bigcup X_1 \cap \bigcup X_2)$ as well. Hence, R_2 is a possible escape space and there is an answer X_3 of the cops playing on $(G \times n, H \cdot n)$ to the robber's choice of R_2 , etc.

In this way we obtain a winning strategy on (G, H) for $\lfloor \frac{k}{n} \rfloor$ cops. This winning strategy is monotone if the winning strategy on $(G \times n, H \cdot n)$ is monotone: $R_1 \supseteq R_2$ implies $R'_1 = \pi^+(R_1) \supseteq \pi^+(R_2) = R'_2$. $\square$

We conject that $c_{H \cdot n}\text{-hw}(G \times n) = n \cdot (c_H\text{-hw}(G))$ holds as well, but as we do not need this statement, we did not try to prove it.

Corollary 3.2.6 *Let (G, H) be the hypergraph pair from Remark 3.2.3 and let $n > 0$ be an integer. Then*

1. $c_{H \cdot n}\text{-cw}(G \times n) = n$, and
2. $c_{H \cdot n}\text{-ghw}(G \times n) = 2n$.

3.2.2 Implementable hypergraph pairs

Definition 3.2.7 *We say that a hypergraph pair (G, H) is trivial, if $\underline{H} \subseteq \underline{G}$.*

Theorem 3.2.8 *Let (G, H) be a trivial hypergraph pair. Then (G, H) can be implemented by a hypergraph $J_{(G, H)}$, i. e. there is a hypergraph $J_{(G, H)}$ such that*

1. $c_H\text{-cw}(G) = c_{J_{(G, H)}}\text{-cw}(J_{(G, H)})$,

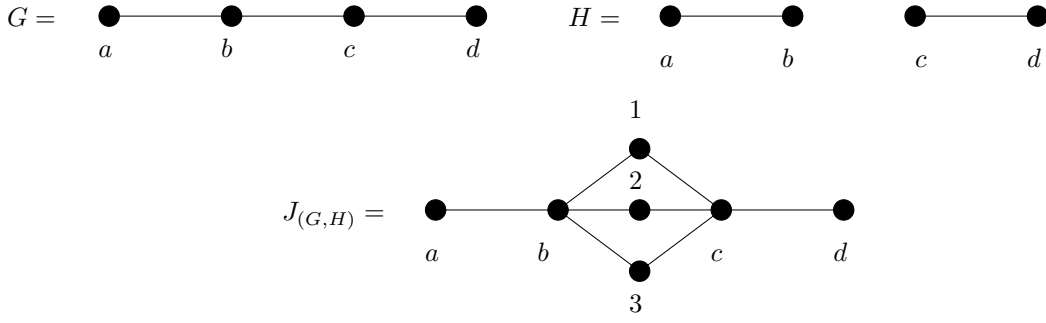


Figure 3.6: A trivial hypergraph pair (G, H) with its implementation $J_{(G,H)}$ as a hypergraph. (In addition to the edges shown in the figure, in each of the three hypergraphs shown there is a singleton edge for each of the four vertices a, b, c, d .) Here the robber edge $\{b, c\}$ is implemented by three parallel paths of length two, because two cops can win monotonely on (G, H) .

$$2. \quad c_H - \text{ghw}(G) = c_{J_{(G,H)}} - \text{ghw}(J_{(G,H)}),$$

$$3. \quad c_H - \text{hw}(G) = c_{J_{(G,H)}} - \text{hw}(J_{(G,H)}).$$

Proof. In a first step, we assume that $\{\{v\} \mid v \in V(G)\} \subseteq E(H)$. Let $n > c_H - \text{hw}(G)$. Define the hypergraph $J_{(G,H)}$ as follows.

Hypergraph $J_{(G,H)}$

vertices: v , where $v \in V(G)$

$(e, 1), \dots, (e, n)$, where $e \in E(\underline{G}) \setminus E(\underline{H})$

edges: h , where $h \in E(H)$

$\{v, (e, i)\}$, where $v \in V(G), e \in E(\underline{G}) \setminus E(\underline{H}), 1 \leq i \leq n$, and $v \in e$

Intuitively, we have implemented a ‘robber edge’ $\{u, v\} \in E(\underline{G}) \setminus E(\underline{H})$ (i. e. an edge which the cops cannot use) in $J_{(G,H)}$ by so many parallel paths of length two between u and v , that the cops cannot catch the robber on the parallel paths without first covering u and v . For a simple example see Figure 3.6.

1: We first show that $c_H - \text{cw}(G) \geq c_{J_{(G,H)}} - \text{cw}(J_{(G,H)})$. Suppose that k cops can win on G . For winning on $J_{(G,H)}$, as long as the robber’s escape space R contains a vertex of $V(G)$, the cops play according to their winning strategy on G . If $R \cap V(G) = \emptyset$, then $R = \{(e, i)\}$ for some $(e, i) \in V(J_{(G,H)})$. Then the robber can be caught by one cop in the next step.

For $c_H - \text{cw}(G) \leq c_{J_{(G,H)}} - \text{cw}(J_{(G,H)})$, assume that k cops have a winning strategy on $J_{(G,H)}$. Note that since we assume that $\{\{v\} \mid v \in V(G)\} \subseteq E(H)$, the hypergraph H is an induced subhypergraph of $J_{(G,H)}$. The cops can win on G by playing according to the ‘induced’ strategy. They win because for chasing the robber through an implementation in $J_{(G,H)}$ of a cop edge $\{u, v\} \in E(\underline{G}) \setminus E(\underline{H})$, they have to cover both u and v . In this way they can also chase the robber through the robber edge $\{u, v\}$.

For showing 2, we can equivalently show that

$$(c_H)^{\text{mon}} - \text{cw}_{\text{mon}}(G) = c_{J_{(G,H)}}^{\text{mon}} - \text{cw}_{\text{mon}}(J_{(G,H)}).$$

This is done similarly to the proof of 1. Note that monotonicity is maintained. 3 is proved like 2.

In a second step, if the hypergraph H does not satisfy $\{\{v\} \mid v \in V(G)\} \subseteq E(H)$, then we can define a hypergraph pair (G', H') by replacing, for an integer $N > c_H - \text{hw}(G)$, each vertex $v \in V(G)$ by an N -clique K_v , as follows.

Hypergraph pair (G', H')

vertices: v' , where $v' \in V(K_v), v \in V(G)$

edges of G' : $\bigcup_{v \in e} V(K_v)$, where $e \in E(H)$
 $\{v'\}$, where $v' \in V(K_v)$ for some $v \in V(G)$
 $\{u', v'\}$, where $u' \in V(K_u), v' \in V(K_v)$ for some $\{u, v\} \in E(\underline{G})$

edges of H' : $\bigcup_{v \in e} V(K_v)$, where $e \in E(H)$
 $\{v'\}$, where $v' \in V(K_v)$ for some $v \in V(G)$

Then (G', H') is again trivial, and we have

1. $c_H - \text{cw}(G) = c_{H'} - \text{cw}(G') = c_{J_{(G', H')}} - \text{cw}(J_{(G', H')})$,
2. $c_H - \text{ghw}(G) = c_{H'} - \text{ghw}(G') = c_{J_{(G', H')}} - \text{ghw}(J_{(G', H')})$,
3. $c_H - \text{hw}(G) = c_{H'} - \text{hw}(G') = c_{J_{(G', H')}} - \text{hw}(J_{(G', H')})$.

Intuitively, this follows from the fact that the singleton edges in H' cannot help the cops to win, because they do not interrupt any connections. $\square$

It seems plausible that for trivial hypergraph pairs, equalities similar to those of Theorem 3.2.8 hold for the other invariants discussed in Section 2.4.

Corollary 3.2.9 *Let (G, H) be a trivial hypergraph pair. Then*

$$c_H - \text{cw}(G) \leq c_H - \text{ghw}(G) \leq c_H - \text{cw}_{\text{mon}}(G) = c_H - \text{hw}(G) \leq 3 \cdot c_H - \text{hw}(G) + 1.$$

Proof. By [AGG05] this is true for the implementing hypergraph $J_{(G, H)}$ of (G, H) , and thus by Theorem 3.2.8 for (G, H) as well. $\square$

3.2.3 Hypergraphs

[Ad04] contains an example of a graph G satisfying

$$c_G - \text{cw}(G) = 4 \text{ and } c_G - \text{ghw}(G) = 5.$$

We now show that we can modify G to obtain a hypergraph G' with $c_{G'} - \text{cw}(G') = 4n$ and $c_{G'} - \text{ghw}(G') = 5n$.

Remark 3.2.10 *If (G, H) is a trivial hypergraph pair, then $(G \times n, H \cdot n)$ is also trivial.* $\square$

Corollary 3.2.11 *Let $n \geq 0$ be an integer. There is a hypergraph G' satisfying*

1. $c_{G'} - \text{cw}(G') = 4n$, and
2. $c_{G'} - \text{ghw}(G') = 5n$.

Proof. Apply Remark 3.2.10 and Theorem 3.2.8 to the graph G from [Ad04] satisfying $c_G - \text{cw}(G) = 4$ and $c_G - \text{ghw}(G) = 5$. $\square$

3.3 The relation of $c_H - \text{ghw}$ and $c_H - \text{hw}$

Recall that by Fact 3.1.9, $c_H - \text{ghw}$ and $c_H - \text{hw}$ are linearly coherent on hypergraphs. With this, in this section we will see:

- $c_H - \text{ghw}$ and $c_H - \text{hw}$ for hypergraphs are linearly coherent, but not strongly coherent invariants.
- The invariants $c_H - \text{ghw}$ and $c_H - \text{hw}$ for hypergraph pairs are not coherent.

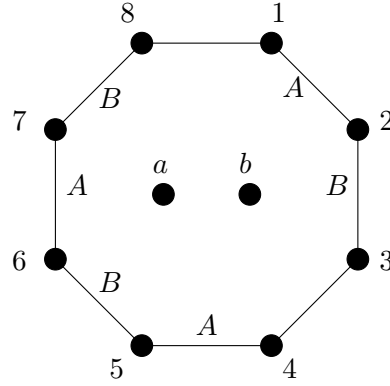


Figure 3.7: The hypergraph H from Proposition 3.3.2. Not shown: Every edge labelled A actually contains the vertex a as well, and every edge labelled B actually contains the vertex b as well.

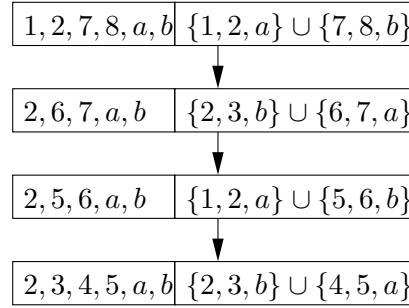


Figure 3.8: A width 2 generalised hypertree decomposition for the hypergraph H from Proposition 3.3.2.

3.3.1 Hypergraphs

In [Ad04, Theorem 4.1] (see also [Ad02]) it was shown that for every $n < \omega$ there is a finite hypergraph H such that $c_H\text{-hw}(H) = c_H\text{-ghw}(H) + n$. In fact it is not hard to check that the hypergraphs constructed in the proof satisfy $c_H\text{-cw}(H) = c_H\text{-ghw}(H)$.

Fact 3.3.1 *For every integer $n \geq 1$ there is a finite hypergraph H such that*

$$c_H\text{-hw}(H) = c_H\text{-ghw}(H) + n.$$

In this section we give a simple example H with

$$c_H\text{-ghw}(H) = 2 \text{ and } c_H\text{-hw}(H) = 3.$$

Proposition 3.4.2 below shows that 2 is in fact the minimal value of $c_H\text{-ghw}(H)$ for any such example. Moreover, we will use this example to prove a stronger version of the above fact.

Proposition 3.3.2 *Consider the following hypergraph H (cf. Figure 3.7).*

Hypergraph H	
vertices:	$1, 2, 3, 4, 5, 6, 7, 8, a, b$
edges:	$\{1, 8\}, \{3, 4\}$
	$\{1, 2, a\}, \{4, 5, a\}, \{6, 7, a\}$
	$\{2, 3, b\}, \{5, 6, b\}, \{7, 8, b\}$

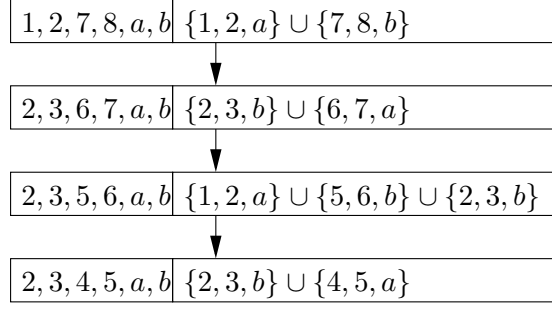


Figure 3.9: A width 3 hypertree decomposition for the hypergraph H from Proposition 3.3.2.

This hypergraph satisfies $c_H\text{-ghw}(H) = 2$ and $c_H\text{-hw}(H) = 3$.

Proof. Figure 3.8 shows a width 2 generalised hypertree decomposition of H . (Note that the second node does not satisfy condition 3 of a hypertree decomposition.) With this it is easy to see that $c_H\text{-ghw}(H) = 2$. Figure 3.9 shows a width 3 hypertree decomposition of H . Thus it remains to show that $c_H\text{-hw}(H) \geq 3$. For the proof we use the game theoretic characterisation of $f\text{-hw}$, Theorem 2.2.12, and Lemmas 3.3.3 and 3.3.4 below. $\square$

We write $A = \{\{1, 2, a\}, \{4, 5, a\}, \{6, 7, a\}\}$ and $B = \{\{2, 3, b\}, \{5, 6, b\}, \{7, 8, b\}\}$ for the subsets of $E(H)$ consisting of those edges containing a or b , respectively. Without restriction the cops always occupy an A edge and a B edge. More precisely:

Lemma 3.3.3 *Let $\bigcup M$, where $M \subseteq E(H)$ and $|M| \leq 2$, be the position of the cops, let A and B be as above, and suppose the cops move to position $\bigcup M'$, where $M' \subseteq E(H)$ and $|M'| \leq 2$.*

- *If $(M \cap A = \emptyset \text{ or } M \cap B = \emptyset)$ and $(M' \cap A = \emptyset \text{ or } M' \cap B = \emptyset)$, then the robber can stay safely on $\{a, b\}$.*
- *If $(M \cap A = \emptyset \text{ or } M \cap B = \emptyset)$ and $M' \cap A \neq \emptyset$ and $M' \cap B \neq \emptyset$, then the robber can reach every vertex of $H \setminus \bigcup M'$, so the cops could just as well have started the game on $\bigcup M'$.*
- *If $M \cap A \neq \emptyset$ and $M \cap B \neq \emptyset$ and $(M' \cap A = \emptyset \text{ or } M' \cap B = \emptyset)$, then the robber can reach a or b and thus the robber wins (because the cops have made an illegal move).*

Lemma 3.3.4 *There is no winning strategy for the cops in $\text{RC}_{\text{mon}}(H, c_H, 2)$.*

Proof. By Lemma 3.3.3 we may assume that the cops always occupy an A edge and a B edge. In the 1st move, the cops choose $M \subseteq E$.

- If $M = \{\{1, 2, a\}, \{2, 3, b\}\}$ then the robber stays safely in $\{5, 6, 7, 8\}$, because the cops can only make monotone moves.
- If $M \subseteq \{\{4, 5, a\}, \{5, 6, b\}, \{6, 7, a\}, \{7, 8, b\}\}$ then the robber stays safely in $\{1, 2, 3\}$, because the cops can only make monotone moves.
- If $M \cap \{\{1, 2, a\}, \{2, 3, b\}\} \neq \emptyset$ and

$$M \cap \{\{4, 5, a\}, \{5, 6, b\}, \{6, 7, a\}, \{7, 8, b\}\} \neq \emptyset$$

- and (w. l. o. g.) $\{1, 2, a\} \in M$, then $\{5, 6, b\} \in M$ or $\{7, 8, b\} \in M$.
- If $\{5, 6, b\} \in M$ the robber stays safely in $\{7, 8\}$.
 - If $\{7, 8, b\} \in M$ the robber stays safely in $\{4, 5\}$.

Thus the robber can escape. $\square$

3.3.2 Ultramonotonicity

In this section all graphs and hypergraphs are finite. We define a new variant of the robber and cops game on G , the ultramonotone robber and cops game on G .

Definition 3.3.5 Let G be a graph, let f be a width function on G , and $k \in \mathbb{R}$. The ultramonotone robber and cops game on G , $\text{RC}_{\text{ultra}}(G, f, k)$, is played like $\text{RC}(G, f, k)$, with the only difference that once a cop is placed on a vertex of G , he cannot be removed again. A winning strategy for the cops in the ultramonotone robber and cops game is defined as in the robber and cops game. In other words: plays $(X_0, r_0, X_1, \dots)$ must satisfy $X_0 \subseteq X_1 \subseteq X_2 \subseteq \dots$ in addition to (R1), (R2) and (C1) ((C2) easily follows from this condition).

Definition 3.3.6 The ultramonotone cop-width of a graph G , $f\text{-cw}_{\text{ultra}}(G)$, is the least number of cops that have a winning strategy in the ultramonotone robber and cops game on G . (Recall that we consider the finite case only).

Remark 3.3.7 For any width function f on a graph G we have

$$f\text{-cw}_{\text{mon}}(G) \leq f\text{-cw}_{\text{ultra}}(G).$$

$\square$

Now we show that for a hypergraph pair (G, H) and the width function c_H , the ultramonotone Robber and Cops Game can be simulated by the monotone robber and cops game in the following sense: We modify (G, H) in such a way, that once a cop has occupied a vertex he cannot release it during the rest of the game without violating monotonicity. Towards this aim, we first show how to implement for given vertices $w, x \in V(G)$ the requirement that once a cop occupies a vertex x it cannot be released as long as w is in the robber's escape space.

Definition 3.3.8 Let G be a graph, let f be a width function on G , let $k \in \mathbb{R}$, and fix $w, x \in V(G)$.

- We call x the watched vertex and w the watch vertex. The watch game on G , $\text{Cops}_{\text{watch}}(w, x)(G, f, k)$, is played like $\text{RC}_{\text{mon}}(G, f, k)$, with the only difference that as long as the watch vertex w is in the robber's escape space, a cop having occupied the watched vertex x is not allowed to release x . (The robber 'watches' the cops' activities on x .)
- The f -watch-width of G is

$$f\text{-watchcw}_{(w, x)}(G) = \inf\{k \in \mathbb{R} \mid \text{the cops have a winning strategy in the watch game on } G \text{ of } f\text{-width } k\}.$$

Remark 3.3.9 Let f be a width function on the graph G and let $w, x \in V(G)$. Then

$$f\text{-cw}_{\text{mon}}(G) \leq f\text{-watchcw}_{(w, x)}(G) \leq f\text{-cw}_{\text{ultra}}(G).$$

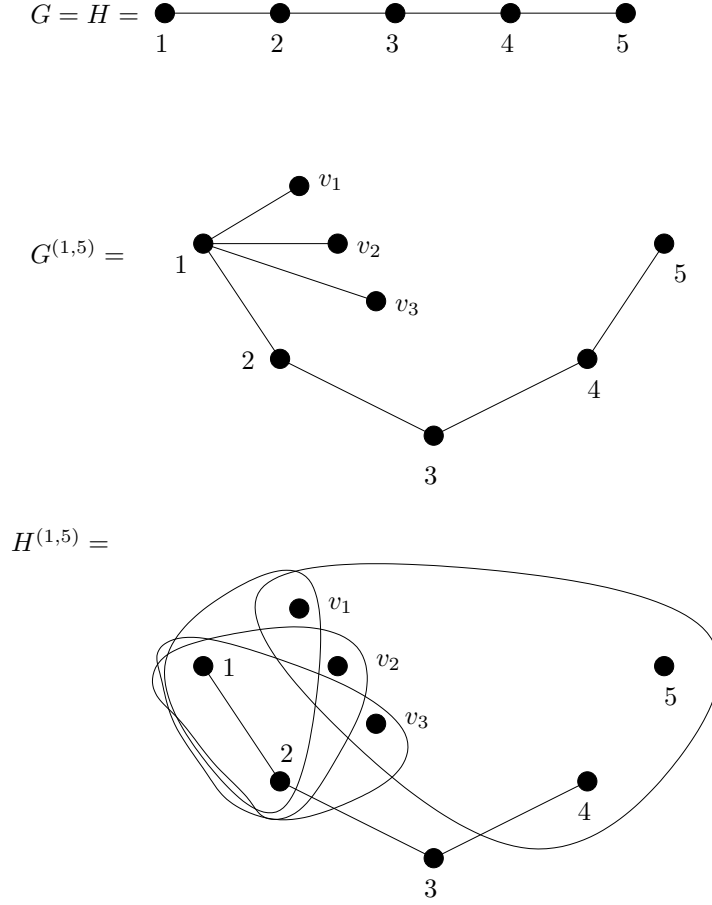


Figure 3.10: A hypergraph pair (G, H) with $G = H$ and the corresponding hypergraph pair $(G^{(1,5)}, H^{(1,5)})$ below.

The proof is immediate from the definitions. $\square$

Now we construct the implementation of the watch game for a hypergraph pair (G, H) (see Theorem 3.3.13 for the precise meaning of this intuition).

Definition 3.3.10 Let (G, H) be a hypergraph pair, $w, x \in V(G)$, and let

$$n = c_H - \text{cw}_{\text{ultra}}(G) + 1.$$

The hypergraph pair $(G^{(w,x)}, H^{(w,x)})$ is defined as follows.

Hypergraph pair $(G^{(w,x)}, H^{(w,x)})$	
vertices:	v , where $v \in V(G)$ $v_1, \dots, v_n$ (new vertices)
edges of $G^{(w,x)}$:	e , where $e \in E(G)$ $\{w, v_1\}, \dots, \{w, v_n\}$
edges of $H^{(w,x)}$:	h , where $h \in E(H)$ and $x \notin h$ $h \cup \{v_1, \dots, v_n\}$, where $h \in E(H)$ and $x \in h$ $h \cup \{v_1\}, \dots, h \cup \{v_n\}$, where $h \in E(H)$ and $w \in h$

Note that (G, H) is an induced subhypergraph pair of $(G^{(w,x)}, H^{(w,x)})$.

Example 3.3.11 Consider the following hypergraph pair (G, H) with $G = H$ (cf. Figure 3.10).

Hypergraph pair (G, H)	
vertices:	1, 2, 3, 4, 5
edges of G :	$\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}$
edges of H :	$\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}$

Obviously, $3 = c_H - \text{cw}_{\text{ultra}}(G) + 1$. Then for $w := 1$ and $x := 5$, we get the following hypergraph pair $(G^{(w,x)}, H^{(w,x)}) = (G^{(1,5)}, H^{(1,5)})$

Hypergraph pair $(G^{(1,5)}, H^{(1,5)})$	
vertices:	1, 2, 3, 4, 5 v_1, v_2, v_3
edges of $G^{(1,5)}$:	$\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}$ $\{1, v_1\}, \{1, v_2\}, \{1, v_3\}$
edges of $H^{(1,5)}$:	$\{1, 2\}, \{2, 3\}, \{3, 4\}$ $\{4, 5, v_1, v_2, v_3\}$ $\{1, 2, v_1\}, \{1, 2, v_2\}, \{1, 2, v_3\}$

Lemma 3.3.12 (Extension by pins) Let (G, H) be a subhypergraph pair of (G', H') . Assume that G and G' are graphs. Let G be an induced subgraph of G' and H and induced subhypergraph of H' . Suppose $V(G') = V(G) \dot{\cup} P$. In addition we require

- (a) For each vertex $p \in P$ there is one and only one vertex $v_p \in V(G)$ such that $\{p, v_p\} \in E(G')$,
- (b) $\{p, v_p\}$ is the only edge of G' containing p , and
- (c) For each $p \in P$ there is a hyperedge $e_p \in E(H')$ such that $\{p, v_p\} \subseteq e_p$.

Then

- 1. $c_H - \text{cw}(G) \geq c_{H'} - \text{cw}(G')$, and
- 2. $c_H - \text{ghw}(G) \geq c_{H'} - \text{ghw}(G')$.

Intuitively, Conditions (a) and (b) express that G' is G with some pins stuck into its vertices.

Proof. For the first part, choose a map $\iota : E(H) \rightarrow E(H')$ satisfying $e \subseteq \iota(e)$ for all $e \in E(H)$. Suppose the cops have a winning strategy for $\text{RC}(G, c_H, k)$. Then they can win $\text{RC}(G', c_{H'}, k)$ as follows:

As long as the robber's escape space contains a vertex $v \in V(G)$, the cops move essentially according to their winning strategy for $\text{RC}(G, c_H, k)$. I.e. if according to their original winning strategy they should move to a set X , then $X = \bigcup Y$ for a set $Y \subseteq E(H)$ of at most k hyperedges of H , and the cops occupy $\bigcup \iota(Y)$ for one such set Y .

If the robber's escape space does not contain any vertex $v \in V(G)$, then it consists of one single vertex $p \in P$ because of (b). Then the robber is caught by one cop moving to e_p , which is possible by (c).

For the second statement, we use Proposition 2.1.21 relating $f^{\text{mon}}\text{-hw}$ and $f\text{-ghw}$ and the game theoretic characterisation of $f^{\text{mon}}\text{-hw}$ in Theorem 2.2.12. It is sufficient to show that

$$(c_H)^{\text{mon}} - \text{cw}_{\text{mon}}(G) \geq (c_{H'})^{\text{mon}} - \text{cw}_{\text{mon}}(G').$$

Suppose the cops have a winning strategy for $\text{RC}_{\text{mon}}(G, (c_H)^{\text{mon}}, k)$. Then a winning strategy for $\text{RC}_{\text{mon}}(G', (c_{H'})^{\text{mon}}, k)$ is as follows: As long as the robber's escape

space contains a vertex $v \in V(G)$, the cops move according to their winning strategy for $\text{RC}_{\text{mon}}(G, (c_H)^{\text{mon}}, k)$. It is easy to see that such a move remains monotone. If the robber's escape space does not contain any vertex $v \in V(G)$, then because of (a) it consists of one single vertex $p \in P$. There the robber is caught by the cops moving to e_p . $\square$

Note that (G, H) and $(G^{(w,x)}, H^{(w,x)})$ always satisfy the assumptions of Lemma 3.3.12.

Theorem 3.3.13 *For every hypergraph pair (G, H) , with $w, x \in V(G)$, the hypergraph pair $(G^{(w,x)}, H^{(w,x)})$ satisfies*

1. $c_H\text{-cw}(G) = c_{H^{(w,x)}}\text{-cw}(G^{(w,x)})$,
2. $c_H\text{-ghw}(G) = c_{H^{(w,x)}}\text{-ghw}(G^{(w,x)})$, and
3. $c_H\text{-watchcw}_{(w,x)}(G) = c_{H^{(w,x)}}\text{-cw}_{\text{mon}}(G^{(w,x)})$.

Proof. Since G is a subgraph of $G^{(w,x)}$, with Remark 2.1.14 and Lemma 3.3.12 it is easy to see that the first and the second equality hold. For the third equality, we first show that

$$c_H\text{-watchcw}_{(w,x)}(G) \leq c_{H^{(w,x)}}\text{-cw}_{\text{mon}}(G^{(w,x)}).$$

Suppose the cops have a winning strategy for $\text{RC}_{\text{mon}}(G^{(w,x)}, c_{H^{(w,x)}}, k)$. Then a winning strategy for $\text{Copswatch}_{(w,x)}(G, c_H, k)$ is obtained as follows: The cops play on G as if they were playing on $G^{(w,x)}$, using a covering hyperedge $h \cap V(G) \in E(H)$ if the hyperedge $h \in E(H^{(w,x)})$ was used in the original game. Thus the width does not increase. Suppose that the robber's escape space still contains the watch vertex w and that there is a cop on the watched vertex x . Since any covering hyperedge of x covers $\{v_i \mid i \in [n]\}$, the cops stand on the v_i as well. But each v_i is a neighbour of w , and by the choice of n , any cover of $\{v_i \mid i \in [n]\}$ in $H^{(w,x)}$ of at most k hyperedges covers x as well. Thus the cops cannot leave x before they go to w without violating monotonicity. Hence we obtain a winning strategy for $\text{Copswatch}_{(w,x)}(G, c_H, k)$.

Now we show that

$$c_H\text{-watchcw}_{(w,x)}(G) \geq c_{H^{(w,x)}}\text{-cw}_{\text{mon}}(G^{(w,x)}).$$

Suppose that the cops have a winning strategy for $\text{Copswatch}_{(w,x)}(G, c_H, k)$. With $n = c_H\text{-cw}_{\text{ultra}}(G) + 1 = |V(G^{(w,x)}) \setminus V(G)|$, consider the following map $\epsilon : E(H) \rightarrow E(H^{(w,x)})$.

$$\epsilon(e) := \begin{cases} e \cup \{v_i \mid i < n\}, & x \in e \text{ and} \\ e & \text{otherwise.} \end{cases}$$

The cops can win as follows: As long as the robber's escape space contains a vertex of $V(G)$, the cops move according to their winning strategy for the game $\text{Copswatch}_{(w,x)}(G, c_H, k)$, where a cover uses the hyperedge $\epsilon(h) \in E(H^{(w,x)})$ if in the original game the hyperedge $h \in E(H)$ was used. If the robber's escape space does not contain any vertex of $V(G)$, then it consists of one single vertex $v_i \in V(G^{(w,x)}) \setminus V(G)$. Then the robber is caught by the cops moving from an edge h that covers the (unique) neighbour of v_i to the edge $h \cup \{v_i\}$. $\square$

In analogy to the definition of $(G^{(w,x)}, H^{(w,x)})$, we can define the hypergraph pair $(G^{(w,xx')}, H^{(w,xx')})$ for $w, x, x' \in V(G)$, with the result that in the game $\text{RC}_{\text{mon}}(G^{(w,xx')}, c_{H^{(w,xx')}}(k))$, both vertices x and x' can be watched from w . We can even implement the requirement that every vertex of G can be watched from every other vertex of G :

Definition 3.3.14 Let (G, H) be a hypergraph pair. Let $n > c_H - \text{cw}_{\text{ultra}}(G)$. We define a hypergraph pair (G^I, H^I) as follows.

Hypergraph pair (G^I, H^I)	
vertices:	v , where $v \in V(G)$ $(v, v', 1), \dots, (v, v', n)$, where $v, v' \in V(G)$ (new vertices)
edges of G^I :	e , where $e \in E(G)$ $\{w, (w, x, 1)\}, \dots, \{w, (w, x, n)\}$, where $w, x \in V(G)$
edges of H^I :	$h \cup (V(G) \times h \times \{1, \dots, n\}) \cup \{(w, x, i)\}$, where $h \in E(H)$, $w, x \in V(G)$ and $1 \leq i \leq n$

Theorem 3.3.15 Let (G, H) be a hypergraph pair and $n > c_H - \text{cw}_{\text{ultra}}(G)$. Let (G^I, H^I) be as above. Then

1. $c_H - \text{cw}(G) = c_{H^I} - \text{cw}(G^I)$,
2. $c_H - \text{ghw}(G) = c_{H^I} - \text{ghw}(G^I)$, and
3. $c_H - \text{cw}_{\text{ultra}}(G) = c_{H^I} - \text{cw}_{\text{mon}}(G^I)$.

Proof. The proof is analogous to the proof of Theorem 3.3.13. □

3.3.3 Hypergraph pairs

Proposition 3.3.16 Let $k < \omega$ be an integer and let G_k be the path on $2^k - 1$ vertices.

Graph G_k	
vertices:	$1, 2, \dots, 2^k - 1$
edges:	$\{1, 2\}, \{2, 3\}, \dots, \{2^k - 2, 2^k - 1\}$

Then

1. $c_{G_k} - \text{cw}(G_k) = c_{G_k} - \text{cw}_{\text{mon}}(G_k) = 1$, and
2. $c_{G_k} - \text{cw}_{\text{ultra}}(G_k) = k$.

Proof. Obviously $c_{G_k} - \text{cw}(G_k) = c_{G_k} - \text{cw}_{\text{mon}}(G_k) = 1$. First we show by induction that the cops have a winning strategy for $\text{RC}_{\text{ultra}}(G_k, c_{G_k}, k)$: For $k = 1$ this is true. Now suppose that the cops have a winning strategy for $\text{RC}_{\text{ultra}}(G_k, c_{G_k}, k)$. Then the cops can win $\text{RC}_{\text{ultra}}(G_{k+1}, c_{G_{k+1}}, k + 1)$ by partitioning the path as follows: The cops occupy the edge $\{2^k, 2^k + 1\}$ dividing the graph into two intervals. Note that $c_{G_k}(\{2^k, 2^k + 1\}) = 1$. Since each of the remaining intervals has at most $2^k - 1$ vertices, for each such interval there is a width k winning strategy by the induction hypothesis.

Now it remains to show that the cops have no winning strategy for the game $\text{RC}_{\text{ultra}}(G_{k+1}, c_{G_{k+1}}, k)$: For $k = 0$ this is true. Now suppose that the cops have no winning strategy for the game $\text{RC}_{\text{ultra}}(G_k, c_{G_k}, k - 1)$. Then the cops have no winning strategy for $\text{RC}_{\text{ultra}}(G_{k+1}, c_{G_{k+1}}, k)$, because for all subsets $X \subseteq V(G_{k+1})$ with $c_{G_{k+1}}(X) \leq 1$ the graph $G_{k+1} \setminus X$ contains a path on at least $2^k - 1$ vertices. By the induction hypothesis, the cops do not have a ultramonotone winning strategy of width $k - 1$ on such a path. □

Corollary 3.3.17 Let $G_k = H_k$ be the path on $2^k - 1$ vertices. Then

1. $c_{H_k^I} - \text{cw}(G_k^I) = 1$,
2. $c_{H_k^I} - \text{ghw}(G_k^I) = 1$, and

$$3. \ c_{H_k^I} - \text{cw}_{\text{mon}}(G_k^I) = k.$$

Proof. By Proposition 3.3.16 we have

$$1 = c_{H_k} - \text{cw}(G_k) \leq c_{H_k} - \text{ghw}(G_k) \leq c_{H_k} - \text{cw}_{\text{mon}}(G_k) = 1,$$

and $c_{H_k} - \text{cw}_{\text{ultra}}(G_k) = k$. Applying Theorem 3.3.13 we obtain

$$c_{H_k^I} - \text{cw}(G_k^I) = c_{H_k^I} - \text{ghw}(G_k^I) = 1,$$

$$\text{and } c_{H_k^I} - \text{cw}_{\text{mon}}(G_k^I) = k. \quad \square$$

Thus we have shown:

Theorem 3.3.18 *For every integer $k \geq 1$ there is a hypergraph pair (G_k, H_k) with $c_{H_k} - \text{ghw}(G_k) = 1$ and $c_{H_k} - \text{hw}(G_k) = k$. $\square$*

3.4 The hypergraph sandwich problem

The *Hypergraph Sandwich Problem* was defined by A. Lustig and O. Shmueli in [LS99]. Proposition 3.4.8 below shows that it is equivalent to deciding for fixed $k < \omega$ whether a given hypergraph pair (G, H) satisfies $c_H - \text{ghw}(G) \leq k$. It is not known whether this problem is solvable in polynomial time. Corollary 3.4.9 shows that it can be approximated in polynomial time when we restrict it to hypergraphs (instead of hypergraph pairs) as input, and Corollary 3.4.10 shows that we can solve it in polynomial time for hypergraph pairs (G, H) where the edge sizes of H are bounded.

3.4.1 Acyclicity

We start with some equivalent formulations of acyclicity. Recall that by Theorem 2.1.5, for a finite hypergraph pair (G, H) there is always a finite generalised hypertree decomposition (T, B, C) satisfying $c_H - \text{width}(T, B, C) = c_H - \text{ghw}(G)$.

Remark 3.4.1 *Let (G, H) be a finite hypergraph. If G has a generalised hypertree decomposition of c_H -width $k \in \omega$, then G also has a generalised hypertree decomposition (T, B, C) of c_H -width k , such that (T, B) is small.*

Proof. Let (T', B', C') be a generalised hypertree decomposition for the finite hypergraph pair (G, H) with finite T' satisfying $c_H - \text{width}(T', B', C') = k = c_H - \text{ghw}(G)$. We may assume that T' is finite. If $\{s, t\} \in E(T')$ with $B'_s \subseteq B'_t$, remove s , B'_s and C'_s from (T', B', C') and connect each neighbour $t' \neq t$ of s by an edge to t . It is easy to see that thus we obtain a new generalised hypertree decomposition (T'', B'', C'') of c_H -width k for G . Since T' was finite, we can repeat this procedure finitely often until we obtain a generalised hypertree decomposition (T, B, C) of c_H -width k for (G, H) s. t. (T, B) is small. $\square$

Note that this is not true for infinite hypergraph pairs: Let $H = (\omega, \mathcal{P}_{<\omega}(\omega))$. Then $c_H - \text{hw}(\underline{H}) = 1$ (take a one way infinite path with an increasing sequence of pieces $B_t = C_t$), but $\underline{H}$ has no small c_H -hypertree decomposition of width 1: Each c_H -hypertree decomposition (T, B, C) of $\underline{H}$ of width 1 satisfies $|B_t| < \omega$ for all $t \in T$. Therefore, for each B_t there is an edge $e \in E(H)$ s. t. $B_t \subsetneq e$ and e is covered by some $t' \in T$.

Recall that a hypergraph H is *acyclic* if its hyperedges can be arranged as nodes of a tree T so that for every vertex $v \in V(H)$, the subgraph of T defined by the nodes containing v is connected.

Proposition 3.4.2 *Let H be a finite tame hypergraph. Equivalent are:*

1. $c_H\text{-ghw}(H) \leq 1$.
2. $c_H\text{-hw}(H) \leq 1$.
3. H is acyclic.

It is easy to see that 3 implies 2, and that 2 implies 1.

$1 \Rightarrow 3$: Let (T, B, C) be a generalised hypertree decomposition of $H = (V, E)$ of c_H -width at most 1. By Remark 3.4.1 we may assume that (T, B) is small. Then $B_t = C_t$ for all $t \in T$. (Otherwise $B_t \subsetneq C_t$ and $C_t = e$ for some $e \in E$. But there must be a node $s \in T$ such that $e \subseteq B_s$. Hence $B_t \subsetneq B_s$, in contradiction to (T, B) being small.) We may assume that $t = B_t$ for every $t \in T$. Now we can extend T to a tree T' with $V(T') = E$ and such that each $e \in E \setminus V(T)$ is attached to some $t \in T$ with $e \in t = B_t$. $\square$

(One could ask which other invariants can be added to this proposition. The 3-clique K_3 , although not acyclic, has $c_{K_3}\text{-link}(K_3) = c_{K_3}\text{-branch-width}(K_3) = 1$. The question whether $c_H\text{-bramble-no}(H) \leq 1$, or at least $c_H\text{-cw}(H) \leq 1$, implies acyclicity is left open.)

Section 4.5 contains another equivalent characterisation.

3.4.2 Hypergraph pairs and the HSP

In this section all hypergraphs are finite.

Definition 3.4.3 *Let (G, H) be a hypergraph pair. G is a projection of H , denoted by $G \leq H$, if for every hyperedge $e \in E(G)$ there exists a hyperedge $e' \in E(H)$ such that $e \subseteq e'$.*

For example, $\underline{H} \leq H$ for every hypergraph H .

Remark 3.4.4 *If the hypergraph pair (G, H) satisfies $G \leq H$, then $E(\underline{G}) \subseteq E(\underline{H})$. If G is a graph, then the converse is also true.*

In the case that G is a graph, the condition $G \leq H$ is in a sense dual to triviality of (G, H) . (Moreover, the method for implementing a trivial hypergraph pair by a hypergraph readily generalises to an implementation of arbitrary hypergraph pairs (G, H) by a hypergraph pair (G', H') such that $G' \leq H'$. But we will not use this.) Hypergraphs of this form feature in the *Hypergraph Sandwich Problem* [LS99]:

HSP
Input: A hypergraph pair (G, H)
Question: Is there an acyclic hypergraph A such that $G \leq A \leq H$?

Intuitively, $G \leq A \leq H$ means that A is ‘sandwiched’ between G and H . This problem clearly is in NP. It is not known whether it is in P.

Lemma 3.4.5 *Let (G, H) be a hypergraph pair. Equivalent are:*

1. *There is an acyclic hypergraph A such that $G \leq A \leq H$.*
2. $c_H\text{-ghw}(G) \leq 1$.

Proof. Let A be acyclic with $G \leq A \leq H$. By the obvious part of Proposition 3.4.2 (which holds even in the infinite case), A has a generalised hypertree decomposition (T, B, C) of width ≤ 1 . For every $t \in T$ we set $C'_t := e'$ where $C_t = e$ and $e \subseteq e' \in E(H)$. Thus we obtain a generalised hypertree decomposition (T, B, C') of (G, H) of width ≤ 1 .

Conversely, let (T, B, C) be a generalised hypertree decomposition of (G, H) of width ≤ 1 . We may assume that $B_t \neq \emptyset$ for all $t \in T$. Define a hypergraph A with $V(A) = V(G)$ and $E(A) = \{B_t \mid t \in T\}$. It is easy to see that A is acyclic. Since every edge $e \in E(G)$ is covered by some B_t we have $G \leq A$. On the other hand, by condition 2 of the definition of a hypertree decomposition of (G, H) , all $t \in T$ satisfy $B_t \subseteq C_t = e'$ for some $e' \in E(H)$. Hence $A \leq H$. $\square$

Since the equivalence of Lemma 3.4.5 is actually a polynomial time equivalence, we can reformulate the hypergraph sandwich problem as follows:

$\text{GHW}^1(\text{PAIR})$
Input: A hypergraph pair (G, H) Question: Is $c_{H\text{-ghw}}(G) \leq 1$?

In Lemma 3.4.7 we will see that we can even reformulate the hypergraph sandwich problem as the problem of deciding whether $c_{H\text{-ghw}}(G) \leq k$ for a fixed integer $k > 0$:

$\text{GHW}^k(\text{PAIR})$
Input: A hypergraph pair (G, H) Question: Is $c_{H\text{-ghw}}(G) \leq k$?

Definition 3.4.6 For a hypergraph H and $k < \omega$, we define the following hypergraph $H^{\leq k}$.

Hypergraph $H^{\leq k}$
vertex set: $V(H)$
edges: $e_1 \cup \dots \cup e_k$, where $e_1, \dots, e_k \in E(H)$

Note that for fixed k and input H , we can compute $H^{\leq k}$ in polynomial time.

Lemma 3.4.7 Let H be a hypergraph. Then:

1. $c_{H\text{-ghw}}(H) \leq k$ if and only if $c_{H^{\leq k}\text{-ghw}}(H) \leq 1$.
2. $c_{H\text{-hw}}(H) \leq k$ if and only if $c_{H^{\leq k}\text{-hw}}(H) \leq 1$.

Proof. It is easy to see that (T, B, C) is a (generalised) hypertree decomposition of $c_{H\text{-width}}$ at most k for H if, and only if, it is a (generalised) hypertree decomposition of $c_{H^{\leq k}\text{-width}}$ at most 1 for H . $\square$

Let GHW^k denote that restriction of $\text{GHW}^k(\text{PAIR})$ to hypergraphs (H, H) as inputs. It is an open problem (stated in [GLS01b]), whether for fixed k , the problem GHW^k is in P.⁴

⁴A recent result is that computing hypertree decompositions is fixed-parameter intractable (see [DF99] for the definitions of the parametrised complexity classes and the corresponding notion of reduction): In [GGMSS05] the authors proved that the problem of deciding whether a hypergraph H satisfies $c_{H\text{-hw}}(H) \leq k$, where k is part of the input, is NP complete and W[2]-hard with respect to parameter k . The same holds for $c_{H\text{-ghw}}(H)$.

Proposition 3.4.8 *Let (G, H) be a hypergraph pair. Equivalent are:*

1. $\text{HSP} \in \text{P}$,
2. $\text{GHW}^1(\text{PAIR}) \in \text{P}$,
3. $\text{GHW}^k(\text{PAIR}) \in \text{P}$.

Furthermore, 3. implies $\text{GHW}^k \in \text{P}$.

Proof. Use Lemmas 3.4.5 and 3.4.7. □

The rest of this section presents two corollaries. The first is that GHW^k can be approximated in polynomial time, and the second is that $\text{GHW}^k(\text{PAIR})$ is decidable in polynomial time for input hypergraph pairs G, H where H is bounded.

From a theoretic point of view, $f\text{-ghw}$ is better behaved than $f\text{-hw}$: It has a simple definition and if f has finite character, it has a compactness property, which $c_H\text{-hw}$ does not have (see Chapter 4).

Corollary 3.4.9 *For a fixed integer k and input a hypergraph H , there is a polynomial time algorithm that correctly returns*

1. $c_H\text{-ghw}(H) > k$, or
2. $c_H\text{-ghw}(H) \leq 3k + 1$ together with a generalised hypertree decomposition of $c_H\text{-width} \leq 3k + 1$, otherwise.

Proof. We can check in polynomial time, whether $c_H\text{-hw}(H) \leq 3k + 1$. If the answer is no, then, by Fact 3.1.9, $c_H\text{-ghw}(H) > k$ and we are in the first case. If the answer is yes, we obtain a hypertree decomposition (see [GLS02]) of $c_H\text{-width} \leq 3 \cdot k + 1$ for H , which obviously also is a generalised hypertree decomposition of $c_H\text{-width} \leq 3k + 1$ for H and we are in the second case. □

By Theorem 3.3.18, this algorithm does not work if we admit hypergraph pairs as input.

If the size of the hyperedges of H of the input hypergraph pair (G, H) is bounded, we can actually decide whether $c_H\text{-ghw}(G) \leq k$ for fixed k .

Corollary 3.4.10 *Fix integers $k, \kappa > 0$. For input (G, H) with H bounded by κ , there is a polynomial algorithm deciding whether $c_H\text{-ghw}(G) \leq k$.*

Proof. Given a hypergraph pair (G, H) , we can compute ΣH in time $O(n^\kappa)$ where $n = |(G, H)|$. Then by Theorem 2.3.16 we can decide in polynomial time whether $c_{\Sigma H}\text{-hw}(G) \leq k$. □

Chapter 4

Compactness

Tree-width is known to be compact in the sense that

$$\text{tw}(G) = \sup \{ \text{tw}(G_0) \mid G_0 \subseteq G, \text{ and } G_0 \text{ is finite} \}.$$

This was first proved by R. Thomas in [Th]. Here we generalise C. Thomassen's short and elegant proof [Thsen89] of this fact to f -ghw. This is where we will need the characterisation of chordal graphs from Section 1.2. Not surprisingly, we need a condition on f for this to work.

Section 4.1 contains a characterisation of f -ghw(G) in terms of triangulations of G with 'small' complete subgraphs, that generalises R. Halin's characterisation (Corollary 1.2.8).

This characterisation is used in Section 4.2 for the proof of the Compactness Theorem¹ 4.2.1. As a corollary we get the extension of a result from [ST93] to infinite graphs G : The cops have a winning strategy for $\text{RC}(G, \text{card}, k)$ if, and only if, they have a winning strategy for $\text{RC}_{\text{mon}}(G, \text{card}, k)$.

In Section 4.3 we show that for a large class of hypergraphs H the invariant c_H -ghw(H) has the compactness property. In Section 4.4 we show that there is an infinite hypergraph H with edge size at most 3 such that $c_H\text{-hw}(H) = 4$, but $c_{H_0}\text{-hw}(H_0) \leq 3$ for all finite induced subhypergraphs H_0 of H . Thus $c_H\text{-hw}(H)$ is not compact.

As an application of the Compactness Theorem 4.2.1, in Section 4.5 another characterisation of f -ghw(G) is presented. For finite graphs and $f = \text{card}$, this characterisation is the k -decomposability as defined by S. Arnborg and A. Proskurowski in [AP86].

4.1 Triangulations

Definition 4.1.1 *Let G be a graph. A monotone width function f on G has finite character if for all infinite $X \subseteq V(G)$ and for all $k \in \mathbb{R}$ there is a finite subset $X_0 \subseteq X$ such that $f(X_0) > k$.*

More generally, a width function f on G (not necessarily monotone) has finite character if f^{mon} has finite character, i. e., if for all infinite $X \subseteq V(G)$ and for all $k \in \mathbb{R}$ there is a finite subset $X_0 \subseteq X$ such that $f(X'_0) > k$ for all finite X'_0 satisfying $X_0 \subseteq X'_0 \subseteq V(G)$.

Example 4.1.2

1. *The width function card has finite character.*

¹In [Ad05a] the Compactness Theorem is proved for the special case of the width function $f = c_H$. The paper also contains the examples from Sections 4.4.

2. Let $c \in \mathbb{R}$, $c > 0$, and let f be a width function on G such that all finite subsets $X_0 \subseteq V(G)$ satisfy $f(X_0) \geq c \cdot |X_0|$. Then f has finite character.

Proof. 1: Obvious. 2: Let $X \subseteq V(G)$ be an infinite subset, and let $k \in \mathbb{R}$. Choose $X_0 \subseteq X$ with $|X_0| > k \cdot \frac{1}{c}$. Then $f(X_0) \geq c \cdot |X_0| > k$. $\square$

Theorem 4.1.3 *Let G be a graph, let f be a monotone width function on G with finite character, and let $k \in \mathbb{R}$. Then*

$$G \text{ has a generalised hypertree decomposition } (T, B, C) \text{ such that} \\ f\text{-width}(T, B, C) \leq k$$

$$\iff$$

G has a triangulation G' such that every complete subgraph K of G' (is finite and) satisfies $f(V(K)) \leq k$.

Proof. ' $\Rightarrow$ ': We may assume that $C = B$. Define G' as follows:

Graph G'

vertex set: $V(G)$

edges: $\{u, v\} \subseteq V(G)$, where $\{u, v\} \subseteq B_t$ for a node $t \in T$.

From Fact 1.2.2 and Theorem 1.2.5, 1, it follows that G is chordal.

Now we show that a complete subgraph of G' cannot be infinite. Otherwise by finite character there exists a finite complete subgraph $K_0 \subseteq K$ with $f(V(K_0)) > k$. Proposition 1.1.8 shows that $V(K_0) \subseteq B_t$ for some $t \in T$. Thus

$$k < f(V(K_0)) \leq f(B_t),$$

a contradiction to $f\text{-width}(T, B, B) \leq k$.

Thus K is finite, $V(K) \subseteq B_t$ for some $t \in T$ (Proposition 1.1.8), and again by monotonicity, we have $f(V(K)) \leq f(B_t) \leq k$.

' $\Leftarrow$ ': Let G' be a triangulation of G such that every complete subgraph $K \subseteq G'$ satisfies $f(V(K)) \leq k$. Then G' is chordal, and using finite character it is easy to see that G' contains no infinite clique. Hence, by Theorem 1.2.5, G' admits a tree decomposition (T, B) into complete pieces. Since all pieces B_t are complete, by assumption all $t \in T$ satisfy $f(B_t) \leq k$. It is easy to see that (T, B, B) is a generalised hypertree decomposition witnessing $f\text{-ghw}(G) \leq k$. $\square$

Note that we have used finite character of f in the proofs of both directions.

Corollary 4.1.4 *Let G be a chordal graph and let f be a monotone width function on G with finite character. Then*

$$f\text{-bramble-no}(G) = f\text{-ghw}(G).$$

Proof. Using Proposition 2.4.8 and Corollary 2.2.20 we get $f\text{-bramble-no}(G) \leq f\text{-ghw}(G)$.

Conversely, if $f\text{-ghw}(G) > k$, then, since G itself is a triangulation of G , by Theorem 4.1.3 there must be a finite complete subgraph K of G such that $f(V(K)) > k$. Then $\mathcal{B} = \{\{v\} \mid v \in V(K)\}$ is a bramble in G with $f\text{-order}(\mathcal{B}) > k$. $\square$

4.2 Compactness of f -ghw

We can now prove the main result of this chapter:

Theorem 4.2.1 (Compactness of f -ghw) *Let G be a graph, let f be a width function with finite character on G , and $k \in \mathbb{R}$. Then*

$$\begin{aligned}
f\text{-ghw}(G) &\leq k \\
&\iff \\
f_{G_0}\text{-ghw}(G_0) &\leq k \text{ for all finite subgraphs } G_0 \text{ of } G.
\end{aligned}$$

Proof. ‘ $\Rightarrow$ ’: This follows from Remark 2.1.14.

‘ $\Leftarrow$ ’: Note that since we may assume f to be monotone (Proposition 2.1.21), $f_{G_0} = f$ for any subgraph G_0 of G (Remark 2.1.17). Suppose now $f\text{-ghw}(G_0) \leq k$ for all finite subgraphs G_0 of G . By Theorem 4.1.3 we may equivalently show that G has a triangulation G' such that every complete subgraph K of G' satisfies $f(V(K)) \leq k$. We prove this using Zorn’s Lemma: Let I be the set of all supergraphs G' of G with $V(G') = V(G)$ that satisfy the following condition.

(*) Every finite subgraph G_0 of G' has a triangulation G'_0 s. t. every complete subgraph K of G'_0 satisfies $f(V(K)) \leq k$.

$I \neq \emptyset$, since $G \in I$: By assumption, $f\text{-ghw}(G_0) \leq k$ for all finite subgraphs G_0 of G . By Theorem 4.1.3, every finite subgraph G_0 of G has a triangulation G'_0 such that every complete subgraph K of G'_0 satisfies $f(V(K)) \leq k$. Hence, $G \in I$.

I is ordered inductively by inclusion: For a transfinite sequence $(G_\alpha)_{\alpha < \delta}$ in I with $G_\alpha \subseteq G_\beta$ for all $\alpha \leq \beta$, $\bigcup_{\alpha < \delta} G_\alpha$ satisfies (*), since a finite subgraph G_0 of $\bigcup_{\alpha < \delta} G_\alpha$ is already contained in some G_{α_0} .

Let $G' \in I$ be a maximal element. The following two claims finish the proof:

Claim (i): Let K be a clique in G' . Then $f(V(K)) \leq k$.

Claim (ii): G' is chordal.

(i): If K is finite, this is true by (*). K cannot be infinite since f has finite character and hence there would be a finite $K_0 \subseteq K$ with $f(V(K_0)) > k$, a contradiction.

(ii): Suppose $O = v_1, \dots, v_m, v_m = v_1$ is an induced cycle of length at least four in G' . By maximality of G' , we cannot add a chord $\{v_i, v_j\}$ to O without producing a finite subgraph G_{ij} of G' such that every triangulation G'_{ij} of G_{ij} contains a complete subgraph K with $f(V(K)) > k$. Let G_0 be the subgraph of G' induced by $(\bigcup_{1 \leq i < j < m} V(G_{ij})) \cup O$. Then G_0 is finite and hence G_0 is contained in some G_{α_1} . But G_0 does not satisfy (*), a contradiction. $\square$

Corollary 4.2.2 (Compactness of tree-width) *Let G be a graph and $k < \omega$. Then*

$$\begin{aligned}
\text{card-ghw}(G) &\leq k \\
&\iff \\
\text{card-ghw}(G_0) &\leq k \text{ for all finite subgraphs } G_0 \text{ of } G.
\end{aligned}$$

Proof. (Recall that by Example 2.1.4, $\text{tw}(G) = \text{card-ghw}(G)$.) The width function card for G has finite character and thus we can apply Theorem 4.2.1. Since card is a monotone width function, by Remark 2.1.17 a finite subgraph G_0 of G satisfies $\text{card}_{G_0}\text{-ghw}(G_0) = \text{card-ghw}(G_0)$. $\square$

Corollary 4.2.3 *Any graph G satisfies*

$$\text{card-bramble-no}(G) = \text{card-cw}(G) = \text{card-cw}_{\text{mon}}(G).$$

Proof. With Proposition 2.4.8 it follows that

$$\text{card-bramble-no}(G) \leq \text{card-cw}(G) \leq \text{card-cw}_{\text{mon}}(G).$$

Conversely, let $\text{card-cw}_{\text{mon}}(G) > k$. Then, using monotonicity of card and the game theoretic characterisation (Theorem 2.2.12), we have $\text{card-ghw}(G) = \text{card-hw}(G) = \text{card-cw}_{\text{mon}}(G) > k$. Thus by Compactness of tree-width, there

is a finite subgraph $G_0 \subseteq G$ satisfying $\text{card-ghw}(G_0) > k$. In [ST93], N. Robertson and P. D. Seymour proved that this implies the existence of a bramble $\mathcal{B}$ in G_0 of $\text{card-order}(\mathcal{B}) > k$. Since this bramble also is a bramble in G , we have $\text{card-bramble-no}(G) > k$. $\square$

4.3 Compactness of c_H -ghw

In general, $c_H\text{-ghw}(H)$ is *not* compact:

Proposition 4.3.1 *The hypergraph $H = (\aleph_1, \mathcal{P}_{<\omega}(\aleph_1))$ satisfies:*

1. $c_{H_0}\text{-ghw}(H_0) \leq 1$ for all finite induced subhypergraphs H_0 of H ,
2. $c_H\text{-ghw}(H) = \infty$.

Proof. 1 is obvious. For 2, recall that $\beta_t = B_{T_t} \setminus B_t^{\text{pred}}$. We show that $\underline{H}$ does not have a tree decomposition at all. Suppose towards a contradiction that (T, B) is a tree decomposition of $\underline{H}$. Then

- (i) Each $t \in T$ has at most one successor s such that $\beta_s \neq \emptyset$.
- (ii) Such a successor s satisfies $B_t \subseteq B_s$.

Proof of (i): Otherwise, there are (at least) two successors s and s' of t with $v \in \beta_s$ and $v' \in \beta_{s'}$. By connectedness, $v \neq v'$. But the edge $\{v, v'\} \in E(\underline{H})$ has to be covered in some node of T , a contradiction to connectedness.

Proof of (ii): Let s be a successor of t with $u \in \beta_s$. Suppose $v \in B_t \setminus B_s$. Again, the edge $\{u, v\}$ has to be covered in some node of T , a contradiction to connectedness.

Together, (i) and (ii) show that $V(H) = \aleph_1$ can be obtained as a countable union of an increasing sequence of finite subsets B_t of $V(H)$, a contradiction. $\square$

Definition 4.3.2 *A hypergraph H is locally bounded, if for all $v \in V(H)$ there is an integer $\kappa_v \geq 0$ such that $\max \{ |e| \mid v \in e \} \leq \kappa_v$.*

Recall that H is bounded, if there is an integer $\kappa \geq 0$ such that all $e \in E(H)$ satisfy $|e| \leq \kappa$. Obviously, if H is bounded, then H is locally bounded.

Proposition 4.3.3 *Let (G, H) be a hypergraph pair. If H is locally bounded, then c_H^{mon} has finite character.*

Proof. If (G, H) is finite, there is nothing to show. Otherwise, fix an integer $k \geq 0$ and let $X \subseteq V(G)$ be an infinite set. Choose a finite set $X_0 \subseteq X$. If $c_H^{\text{mon}}(X_0) > k$, then we are finished. Otherwise choose a finite set $X_1 \subseteq X \setminus X_0$ satisfying $|X_1| \geq \max \{ \kappa_v \mid v \in X_0 \}$. Then $c_H^{\text{mon}}(X_0 \cup X_1) > c_H^{\text{mon}}(X_0)$, hence $c_H^{\text{mon}}(X_0 \cup X_1) \geq c_H^{\text{mon}}(X_0) + 1$. Again, if $c_H^{\text{mon}}(X_0) > k$, then we are finished. Otherwise we continue in the same way until we get $c_H^{\text{mon}}(X_0 \cup \dots \cup X_n) > k$. $\square$

The following example shows that the converse is not true.

Example 4.3.4 *Let H be the following hypergraph:*

Hypergraph H	
vertices:	$a,$ $(n, 1), \dots, (n, n), \text{ where } n < \omega$
edges:	$\{a, (n, 1), \dots, (n, n)\}, \text{ where } n < \omega$

Obviously, c_H^{mon} has finite character, but the sizes of the hyperedges containing the vertex a are unbounded.

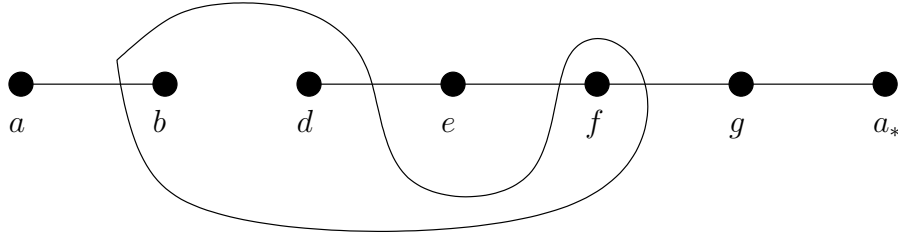


Figure 4.1: The hypergraph pair (G', H') from the proof of Theorem 4.4.1.

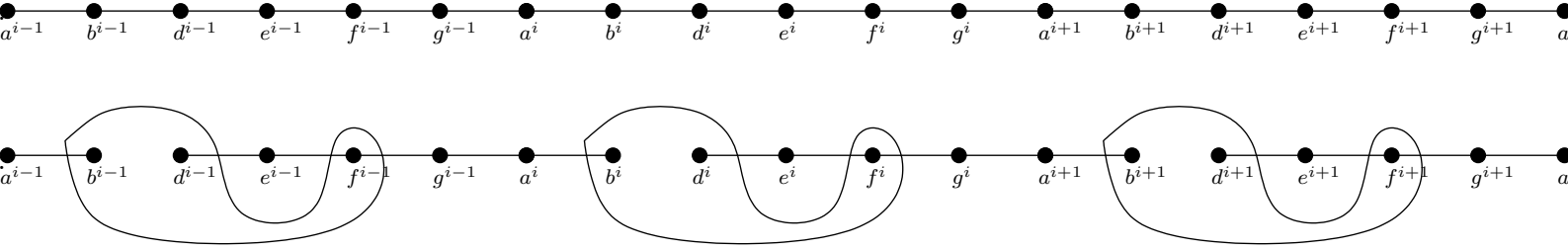


Figure 4.2: The hypergraph pair (G, H) from Theorem 4.4.1.

Corollary 4.3.5 *Let (G, H) be a hypergraph pair where H is locally bounded. Then*

$$\text{c}_H\text{-ghw}(G) \leq k$$

$$\iff$$

Proof. Use Theorem 4.2.1.

4.4 Non-compactness of c_H -hw

4.4.1 Hypergraph pairs

In this section, we show:

Theorem 4.4.1 *There is a hypergraph pair (G, H) such that:*

- (i) $c_H\text{-hw}(G) = 2$,
- (ii) $c_{H_0}\text{-hw}(G_0) \leq 1$ for all finite induced subhypergraph pairs (G_0, H_0) of (G, H) ,
and
- (iii) all hyperedges $e \in E(H)$ satisfy $|e| \leq 3$.

The rest of this section consists of the proof of this Theorem:

Let (G', H') be the hypergraph pair defined by $V(G') = \{a, b, d, e, f, g, a_*\}$,

$$E(G') = \{\{a, b\}, \{b, d\}, \{d, e\}, \{e, f\}, \{f, g\}, \{g, a_*\}\} \text{ and}$$
$$E(H') = \{\{a, b\}, \{b, d, f\}, \{d, e\}, \{e, f\}, \{f, g\}, \{g, a_*\}\} \text{ (see Figure 4.1).}$$

For each $i \in \mathbb{Z}$, let (G^i, H^i) be an isomorphic copy of (G', H') with vertex set

$\{a^i, b^i, d^i, e^i, f^i, g^i, h^i\}$. By identifying the ‘rightmost’ vertex of the i -th copy with the first vertex of the $(i+1)$ -th copy, we obtain the hypergraph pair of Figure 4.2

$$(G, H) := \bigcup_{i \in \mathbb{Z}} (G^i, H^i) / (a_*^i = a^{i+1}).$$

The game theoretic characterisation of $c_H\text{-hw}(G)$ (Theorem 2.2.12) allows us to argue game theoretically: It is easy to see that two cops have a winning strategy on (G, H) , and we will not prove this here.

Let (G_0, H_0) be a finite induced subhypergraph pair of (G, H) . Then one cop player has a winning strategy: He starts by occupying the rightmost hyperedge of (G_0, H_0) and then he chases the robber to the left.

Nevertheless, one cop does not have a winning strategy on (G, H) . The robber can escape as follows: After the cop’s first move to a hyperedge $h \in E(H)$, the robber chooses the unoccupied vertex e^i which is as far ‘left’ as possible such that there is a vertex $v \in V(G)$ on the ‘left side’ of e^i with $v \in h$. From then on, the robber stays on e^i . By monotonicity it is easy to see that each of the cop’s following moves is uniquely determined, until at some point, the cop occupies the hyperedge $\{b^i, d^i, f^i\}$. Now he cannot move at all, and therefore the robber wins. Hence (G, H) satisfies (i), (ii) and (iii).

4.4.2 Hypergraphs

In this section we prove:

Theorem 4.4.2 *There is an infinite hypergraph H satisfying:*

- (i) $c_H\text{-hw}(H) = 4$,
- (ii) $c_{H_0}\text{-hw}(H_0) \leq 3$ for all finite induced subhypergraphs H_0 of H , and
- (iii) all hyperedges $e \in E(H)$ satisfy $|e| \leq 3$.

Although the presentation will be independent from the previous paragraph, the idea of the construction is to code G and H from the previous paragraph in a single hypergraph. So we will have to ‘implement’ the *cop edges* (i. e. the edges from $E(H) \setminus E(G)$) and the *robber edges* (i. e. the edges from $E(G) \setminus E(H)$) using ordinary edges that can be used both by the cops and by the robber. The implementation of the cop edges will increase the number of cops necessary to catch the robber by two.

As in the previous section, we again construct the example by gluing together infinitely many copies of a certain ‘module’: Let M be the hypergraph from Figure 4.3, with $V(M) = P \cup Q \cup \{c_1, c_2, \dots, c_5\}$, where $P = \{\alpha, \beta, \gamma, \alpha', \beta', \gamma'\}$, and $Q = \{a, b, d, e, f, g, a_*\}$,
 $E(M) = \{\{a, b\}, \{d, e\}, \{e, f\}, \{f, g\}, \{g, a_*\}, \{\alpha, \beta, \gamma\}, \{\alpha', \beta', \gamma'\}\}$
 $\cup \{\{b, c_i\} \mid i = 1 \dots, 5\} \cup \{\{c_i, d\} \mid i = 1 \dots, 5\} \cup \{\{\alpha, b, \alpha'\}, \{\beta, d, \beta'\}, \{\gamma, f, \gamma'\}\}$
 $\cup \{\{p, q\} \mid p \in P, q \in Q\} \cup \{\{v\} \mid v \in V(M)\}.$

Let $(M^i)_{i \in \mathbb{Z}}$ be a family of isomorphic copies of M . We denote the vertices of M^i by $\{v^i \mid v \in V(M)\}$. By identifying the rightmost vertex of the i -th copy with the first vertex of the $(i+1)$ -th copy, we obtain the hypergraph

$$H := \bigcup_{i \in \mathbb{Z}} (M^i) / (a_*^i = a^{i+1}).$$

Let $S = \bigcup_{i \in \mathbb{Z}} \{a^i, b^i, c_1^i, d^i, e^i, f^i, g^i, a_*^i\}$.

Proof of Theorem 4.4.2, (i). It is easy to see that four cops can win on H . The following lemma shows that $\text{mon-mw}(H) > 3$:

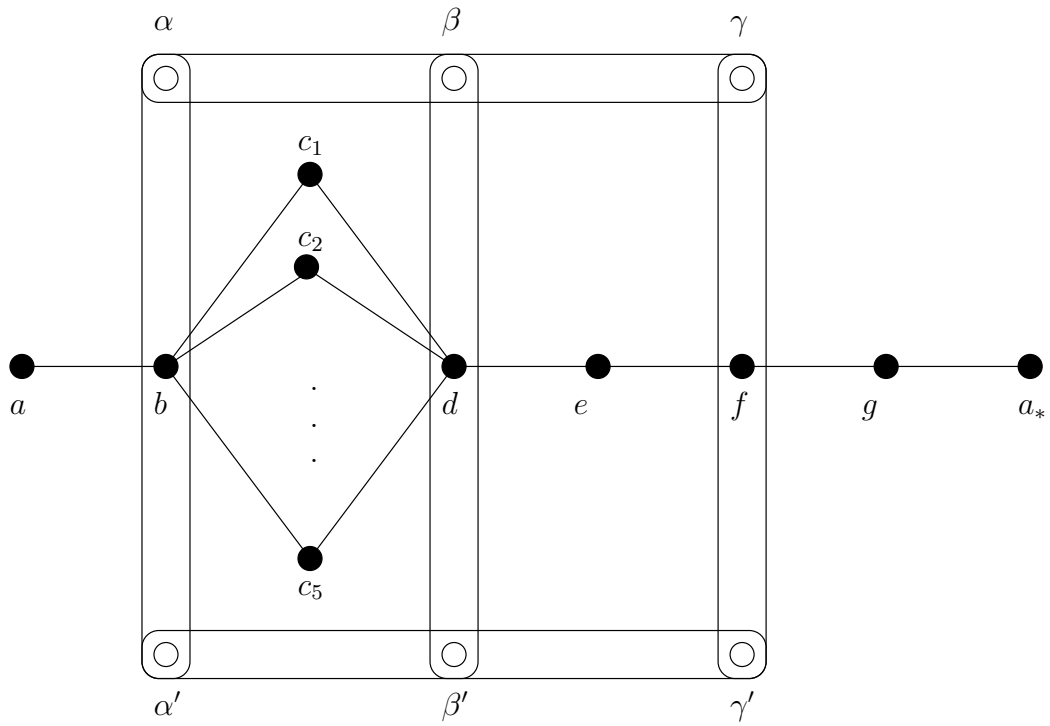


Figure 4.3: The hypergraph M from the proof of Theorem 4.4.2. In addition to the hyperedges shown in the figure, every vertex from $P = \{\alpha, \beta, \gamma, \alpha', \beta', \gamma'\}$ is connected to every vertex from $Q = \{a, b, d, e, f, g, a_*\}$ by a 2-edge. Also, for every $v \in V(M)$, M contains the hyperedge $\{v\}$.

Lemma 4.4.3 *On H as defined above, the robber can win against three cops.*

Proof. We describe such a winning strategy, which is similar to the robber's winning strategy on the hypergraph pair (G, H) defined in the previous paragraph: As long as no vertex from S is occupied by a cop, the robber stays in the connected component of H containing S . Sooner or later, the cops will have to first occupy one or more vertices of S . Let v be the rightmost such. Then the robber reacts by moving to the leftmost vertex a^i that is to the right of v . Let H' be the restriction of H to M^i together with the path from v to a^i (in the sense of induced subhypergraph). The following lemma shows that the robber can elude capture. $\square$

Lemma 4.4.4 *Let N be the following hypergraph.*

Hypergraph N	
vertex set:	$\{v_n, \dots, v_0\} \cup V(M)$
edges:	$\{v_n, v_{n-1}\}, \{v_{n-1}, v_{n-2}\}, \dots, \{v_1, v_0\}, \{v_0, a\}$ $e, \text{ where } e \in E(M)$

The robber can win on N against three cops, if in the first move the cops occupy v_n .

Proof. The robber remains on a until a cop occupies a . Then the robber plays according to his winning strategy from Lemma 4.4.5 below. $\square$

Lemma 4.4.5 *The robber can win against three cops on M , if the first game position is $(\bigcup X, a)$ with some $X \subseteq E(M)$, $|X| \leq 2$ and if in the second move some cop occupies an edge containing the vertex a .*

Proof. As long as there is a vertex from P that is unoccupied by the cops, the robber stays on such a vertex. All this time the cops have to occupy a . As soon as the cops occupy all vertices from P (the cops still have to occupy a), we have one of the following two situations:

- The cops occupy $\{\alpha, \beta, \gamma\}, \{\alpha', \beta', \gamma'\}$ and $\{a\}$, or
- they occupy $\{\alpha, \beta, \gamma\}, \{\alpha', \beta', \gamma'\}$ and $\{a, b\}$.

In the first case the robber stays on b until the cops occupy b , by which we are in the second case. In the second case the robber moves to e . Now the cops can either additionally occupy a vertex c_i (and then give up), or rearrange themselves to occupy $\{\alpha, b, \alpha'\}, \{\beta, d, \beta'\}$ and $\{\gamma, f, \gamma'\}$ (and then give up as well). $\square$

Proof of Theorem 4.4.2, (ii). Three cops have a winning strategy on a finite induced subhypergraph H_0 of H : First, we may assume that H_0 is the union of finitely many 'modules' M_i . The proof idea is that two cops occupy hyperedges of the type $\{\alpha^i, \beta^i, \gamma^i\}$ and $\{\alpha'^i, \beta'^i, \gamma'^i\}$ almost all the time. The third cop chases the robber from right to left, while the other two cops do not change position except when the third cop reaches an edge of type $\{d^i, e^i\}$. Then the next step for the cops is to occupy $\{\alpha^i, b^i, \alpha'^i\}, \{\beta^i, d^i, \beta'^i\}$ and $\{\gamma^i, f^i, \gamma'^i\}$. After that, the robber is either caught on one of the vertices $c_1^i, c_2^i, c_3^i, c_4^i, c_5^i$, or two cops occupy the hyperedges $\{\alpha^{i-1}, \beta^{i-1}, \gamma^{i-1}\}$ and $\{\alpha'^{i-1}, \beta'^{i-1}, \gamma'^{i-1}\}$ and the third cop continues chasing the robber towards the left end of the finite hypergraph, until the robber finally cannot move.

Thus, $c_{H_0}\text{-cw}_{\text{mon}}(H_0) \leq 3$ for all finite induced subhypergraphs H_0 of H . Altogether, H satisfies (i), (ii) and (iii) as required. $\square$

4.5 k -decomposability and f -ghw

As an application of the compactness theorem, we present a characterisation of generalised f -hypertree-width. For finite graphs and $f = \text{card}$, this characterisation is k -decomposability as defined by S. Arnborg and A. Proskurowski in [AP86].

Definition 4.5.1 *Let $k > 0$ be an integer, and let G be a graph equipped with a monotone width function f . The equipped graph (G, f) is k -decomposable, if*

(D1) $f(V(G)) \leq k$, or

(D2) G contains a finite set $S \subseteq V(G)$ such that the connected components of $G \setminus S$ are $(C_\alpha)_{\alpha < \beta}$ (for some suitable ordinal β), and all equipped graphs (G_α, f_{G_α}) defined as follows for $\alpha < \beta$ are k -decomposable:

vertex set:	$C_\alpha \cup S$
edges:	$\{u, v\}$ where $u, v \in S$, and e where $e \in E(G[C_\alpha \cup S])$,

or

(D3) for a limit ordinal λ we have $G = \bigcup_{\alpha < \lambda} G_\alpha$, where $(G_\alpha)_{\alpha < \lambda}$ is a family of induced subgraphs G_α of G with $G_\alpha \subseteq G_\beta$ for $\alpha < \beta < \lambda$, such that all equipped graphs (G_α, f_{G_α}) are k -decomposable.

Remark 4.5.2 *If (G, f) is k -decomposable and $G_0 \subseteq G$ is a subgraph, then (G_0, f_{G_0}) is also k -decomposable.*

Proof. This is easily proved by induction on the rules (D1), (D2), (D3). $\square$

Proposition 4.5.3 *Let G be a finite graph equipped with a monotone width function f .*

$$\begin{aligned} f\text{-ghw}(G) &\leq k \\ &\iff \\ (G, f) &\text{ is } k\text{-decomposable.} \end{aligned}$$

Proof. ‘ $\Rightarrow$ ’: We use induction on $|V(G)|$. Let (T, B, C) be a generalised hypertree decomposition witnessing that $f\text{-ghw}(G) \leq k$. By Remark 3.4.1 we may assume that (T, B) is small. If $f(V(G)) > k$, then $|V(T)| \geq 2$. Let $(s, t) \in E(T)$. Then the set $S := B_s \cap B_t$ separates G , and every component C of $G \setminus S$ satisfies $|C \cup S| < |V(G)|$. By induction hypothesis, $(G[C \cup S], f_{G[C \cup S]})$ is k -decomposable. Thus by (D2), (G, f) is k -decomposable.

‘ $\Leftarrow$ ’: We prove this direction by induction on the rules (D1) and (D2). If $f(V(G)) \leq k$, there is nothing to show. Otherwise, by induction hypothesis, $f_{G_i}\text{-ghw}(G_i) \leq k$ for $i = 1 \dots, m$, witnessed by (T^i, B^i, C^i) . Since S induces a clique in G_i , by Proposition 1.1.8, there exists a node $t_i \in V(T^i)$ such that $S \subseteq B_{t_i}^i$ ($i = 1 \dots, m$). Then (T', B', C') is a generalised hypertree decomposition of G of f -width at most k , where

$$\begin{aligned} V(T') &:= \bigcup V(T^i) \cup \{t\}, \\ E(T') &:= \bigcup E(T^i) \cup \{\{t_i, t\} \mid i = 1, \dots, m\} \\ B'_s &:= B_s^i \text{ for } s \in V(T^i), & B'_t &:= S, \\ C'_s &:= C_s^i \text{ for } s \in V(T^i) & \text{and} & C'_t &:= S. \end{aligned}$$

$\square$

Corollary 4.5.4 *Let $k \in \mathbb{R}$, and let G be a graph equipped with a monotone width function f .*

1. *If $f\text{-ghw}(G) \leq k$, then (G, f) is k -decomposable.*
2. *If f has finite character, then*

$$f\text{-ghw}(G) \leq k \text{ if and only if } (G, f) \text{ is } k\text{-decomposable.}$$

Proof. If G is finite, this is true by Proposition 4.5.3. So let G be infinite.

1: Let G be a graph equipped with a monotone width function f , and let (T, B, C) be a generalised hypertree decomposition of G with $\text{width}(T, B, C) \leq k$. Define the supergraph G' of G as follows:

Graph G'	
vertex set:	$V(G)$
edges:	$\{u, v\}$ where $\{u, v\} \subseteq B_t$ for some node $t \in T$

Then G is a subgraph of G' , f is a monotone width function on G' , and (T, B, C) witnesses that $f\text{-ghw}(G') \leq k$. By Remark 4.5.2, it suffices to show that (G', f) is k -decomposable. We number the nodes of T , beginning with the root t_0 of T in such a way that the numbering respects the predecessor relation. Now we show by induction that for every ordinal α the subgraph G_α of G' induced by $\bigcup_{\beta \leq \alpha} B_{t_\beta}$, together with f_{G_α} is k -decomposable.

Since $f(B_{t_0}) \leq k$, by (D1) this is true for $G_0 = G'[B_{t_0}]$. Suppose that (G_α, f_{G_α}) is k -decomposable. Since $f_{G'[B_{t_{\alpha+1}}]}(B_{t_{\alpha+1}}) \leq k$, it follows from (D1) that the equipped graph $(G'[B_{t_{\alpha+1}}], f_{G'[B_{t_{\alpha+1}}]})$ is k -decomposable. By definition of G' , the set $S := B_{t_{\alpha+1}} \cap B_{\text{pred}(t_{\alpha+1})} \subseteq V(G_\alpha)$ induces a complete subgraph in G_α . Thus we can apply (D2) and it follows that $(G_{\alpha+1}, f_{G_{\alpha+1}})$ is k -decomposable.

Let λ be a limit ordinal. Suppose that for every $\alpha < \lambda$, the equipped graph (G_α, f_{G_α}) is k -decomposable. Then every G_α is an induced subgraph of G_λ , and we have $G_\alpha \subseteq G_\beta$ for $\alpha < \beta < \lambda$, and $G_\lambda = \bigcup_{\alpha < \lambda} G_\alpha$. Thus by (D3), $(G_\lambda, f_{G_\lambda})$ is k -decomposable.

2. Let f have finite character. By 1 it suffices to show that if (G, f) is k -decomposable, then $f\text{-ghw}(G) \leq k$. Let G be (f, k) -decomposable. Then so is every finite subgraph G_0 of G . Hence by Proposition 4.5.3, every finite subgraph G_0 of G satisfies $f\text{-ghw}(G_0) \leq k$. Thus by Theorem 4.2.1, $f\text{-ghw}(G) \leq k$. $\square$

Since rule (D2) can be used to create infinite graphs, one might ask whether rule (D3) is redundant. A *ray* in a graph is a subgraph (not necessarily induced) isomorphic to the graph shown in Figure 2.2, B. It is easy to see that, regardless of f , no graph created only using rules (D1) and (D2) can contain a ray. But a ray equipped with $f = \text{card}$ should, of course, be 2-decomposable. Hence (D3) is in fact necessary.

Chapter 5

More width functions

In this chapter three more examples of width functions are given. In the first two sections, we define two width functions that make use of functional dependencies in databases or structures (see Section 2.3) as follows. Let σ be a signature (possibly containing partial function symbols). Let $\mathcal{M}$ be a σ -structure and $X \subseteq M$. Let $\langle X \rangle$ denote the smallest superset $\langle X \rangle \supseteq X$ that is a universe of a substructure of $\mathcal{M}$ that is closed under partial functions. Then for input $\mathcal{M}$ and $X \subseteq M$ we can compute $\langle X \rangle$ in polynomial time in the size $\|\mathcal{M}\|$. (Starting with X we iterate through the partial functions of σ , adding an element y to X whenever $p(\bar{x}) = y$ for some $\bar{x} \subseteq X$ and $p \in \sigma$. We stop, when after a run through all partial function symbols of σ , no changes have occurred. Thus we stop after at most $|M|$ such runs, each of which can be done in polynomial time.)

To apply Theorem 2.3.13 and Corollary 2.3.17, we need a class $\mathcal{C}_1$ such that we have $\text{SUBSTR}_{\text{enum}}^{\mathcal{C}_1} \in \text{P}$ and $\text{HOM}_{\text{enum}}(\mathcal{C}_1, _) \in \text{P}$; and the closed substructures enumerated by $\text{SUBSTR}_{\text{enum}}^{\mathcal{C}_1}$ should be large, so that the promise and no promise algorithms work on large classes of structures. We can achieve this by taking $\mathcal{C}_1$ to consist of the closures of a bounded number of elements (Section 5.1) or of unions of a bounded number of hyperedges in the underlying hypergraph (Section 5.2).

In Section 5.1 we actually define a width function on a *mixed graph*, i. e. on a graph that may contain directed edges. This width function can be seen as the directed analogue of the function *card*. If all partial function symbols from σ are unary, a σ -structure has a natural underlying mixed graph.

Since every directed graph is a mixed graph, we thus get a new tree-width notion for directed graphs. Up to now, basically two tree-width notions for directed graphs have been defined. The *directed tree-width* was introduced by T. Johnson, N. Robertson, P. D. Seymour, and R. Thomas in [JRST01]. They showed that Hamiltonicity can be solved in polynomial time for directed graphs of bounded directed tree-width¹. However, the definition of directed tree-width is rather complicated and it has some unfavourable properties². Recently, D. Berwanger, A. Dawar, P. Hunter, and S. Kreutzer introduced the notion of *DAG-width* of a directed graph in [BDHK05]. The directed tree-width of a digraph is at most its DAG-width, so Hamiltonicity can also be solved in polynomial time for directed graphs of bounded DAG-width. Moreover, the winner of a *parity game* on a directed graph of bounded DAG-width can be determined in polynomial time. The tree-width notion for directed graphs presented here is fundamentally different from the two previous notions: it is coherent with neither of them.

In Section 5.2 we define a width function on *directed hypergraphs*, which is the

¹Actually, in [Re99] B. Reed gave an alternative definition of directed tree-width, but I could not verify that this definition is equivalent to directed tree-width.

²In [Ad05b] it was proved that taking minors can increase the directed tree-width.

directed analogue for the width function c_H . For an arbitrary signature σ , a σ -structure $\mathcal{M}$ always has an underlying directed hypergraph.

Finally, in Section 5.3 we show that the *fractional hypertree-width* recently defined by M. Grohe and D. Marx in [GM05] can also be described by a width function. It has been asked in [CD05, CJG05, GGMSS05, Gr03] whether there are classes $\mathcal{C}$ of structures whose underlying hypergraphs have unbounded c_H -hypertree-width such that $\text{HOM}(\mathcal{C}, -) \in \text{P}$. In [GM05] this question is answered positively if we take $\mathcal{C}$ to be the class of structures whose underlying hypergraphs have bounded fractional hypertree-width. Applying our general theory we prove some inequalities between ‘fractional’ hypergraph invariants, such as ‘fractional’ bramble-number, ‘fractional’ branch-width, etc.

5.1 Mixed graphs

5.1.1 Definitions and some observations

Definition 5.1.1 A mixed graph is a triple $\vec{G} = (V, E, D) = (V(\vec{G}), E(\vec{G}), D(\vec{G}))$ where $E \subseteq \mathcal{P}_{=2}(V)$ (the set of edges of $\vec{G}$) and $D \subseteq V \times V$ (the set of arcs of $\vec{G}$). Thus (V, E) is a graph and (V, D) is a directed graph.

We identify graphs (V, E) with the mixed graphs $(V, E, \emptyset)$ and directed graphs (V, D) with the mixed graphs $(V, \emptyset, D)$.³ The underlying graph $\underline{\vec{G}}$ of a mixed graph $\vec{G}$ is defined as follows.

Graph $\underline{\vec{G}}$	
vertex set:	$V(\vec{G})$
edges:	e , where $e \in E(\vec{G})$ $\{u, v\}$, where $(u, v) \in D(\vec{G})$

Definition 5.1.2 Let $\vec{G} = (V, E, D)$ be a mixed graph. The forward closure of $X \subseteq V(\vec{G})$ is the smallest superset $\langle X \rangle \subseteq V$ of X which is closed under out-edges, i. e. if $u \in \langle X \rangle$ and $(u, v) \in D(\vec{G})$, then $v \in \langle X \rangle$. For a single vertex $v \in V$ we write $\langle v \rangle$ for $\langle \{v\} \rangle$.

Remark 5.1.3 Let $\vec{G}$ be a mixed graph. Any two vertex sets $X, Y \subseteq V(\vec{G})$ satisfy

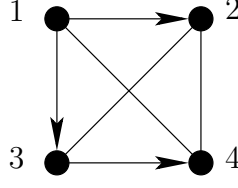
$$\langle X \rangle \cup \langle Y \rangle = \langle X \cup Y \rangle. \quad \square$$

Lemma 5.1.4 Let $\vec{G}$ be a finite mixed graph and let (T, B) be a tree decomposition of the underlying graph $\underline{\vec{G}}$. Then $(T, \langle B \rangle)$ is also a tree decomposition of $\vec{G}$, where $\langle B \rangle := (\langle B_t \rangle)_{t \in T}$.

Proof. Obviously, every vertex and every edge of $\vec{G}$ is covered in $(T, \langle B \rangle)$, since this is the case in (T, B) . Therefore (TD1) and (TD2) hold. Let $(u, v) \in D(\vec{G})$. Both u and v induce connected subtrees T_u and T_v of T . Since $\{u, v\} \in E(\vec{G})$, $\{u, v\}$ is covered somewhere in $T_u \cap T_v$. Therefore adding the vertex v to every block containing the vertex u preserves connectedness of the pieces containing v . By induction, the connectedness condition (TD3) holds as well. Thus, $(T, \langle B \rangle)$ is a tree decomposition of $\vec{G}$. $\square$

Lemma 5.1.4 would stay true for infinite graphs, if we allowed tree decompositions with infinite pieces.

³This is of course inconsistent with the usual identification of graphs with certain directed graphs. We hope that the reader will not find this too confusing.

Figure 5.1: A mixed graph $\vec{G}$ (cf. Example 5.1.6).

By a *width function* for a mixed graph $\vec{G}$ we understand a width function on its underlying graph $\underline{\vec{G}}$.

Definition 5.1.5 For finite mixed graphs $\vec{G}$ we consider the following width function.

$$\begin{aligned} \overrightarrow{\text{card}}_{\vec{G}} : \mathcal{P}_{<\omega}(V(\vec{G})) &\rightarrow \mathbb{R} \cup \{\infty\}, \\ X &\mapsto \inf \{ |Y| \mid Y \subseteq X, X = \langle Y \rangle \}. \end{aligned}$$

Note that $\overrightarrow{\text{card}}_{\vec{G}}$ is not monotone. Also note that defining $\overrightarrow{\text{card}}_{\vec{G}}$ in this way does not make much sense for arbitrary infinite mixed graphs. We write $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G})$ for $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\underline{\vec{G}})$; analogously for the other widths.

Example 5.1.6

- If $\vec{G}$ is a graph (i. e. $D(\vec{G}) = \emptyset$), then $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G}) = \text{card-hw}(\vec{G})$.
- The following mixed graph $\vec{G}$ is depicted in Figure 5.1:

Mixed graph $\vec{G}$	
vertices:	1, 2, 3, 4
edges:	$\{1, 4\}, \{2, 3\}, \{2, 4\}$
arcs:	$(1, 2), (1, 3), (3, 4)$

It satisfies $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G}) = 1$: Take the tree decomposition $(\{t\}, B)$ with $B_t = \{1, 2, 3, 4\}$ and cover B_t by the forward closure of $\{1\}$.

- For an integer $n > 0$ consider the following mixed graph $\vec{K}_n$.

Mixed graph $\vec{K}_n$	
vertices:	$1, 2, \dots, n$
edges:	none
arcs:	(i, j) , where $1 \leq i < j \leq n$

Then $\overrightarrow{\text{card}}_{\vec{K}_n}\text{-hw}(\vec{K}_n) = 1$ and $\text{card-hw}(\vec{K}_n) = n$.

Hence card-hw and $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}$ are not linearly coherent.

Lemma 5.1.7 For every finite mixed graph $\vec{G}$ we have

1. $\overrightarrow{\text{card}}_{\vec{G}}\text{-cw}(\vec{G}) \leq \text{card-cw}(\underline{\vec{G}})$,
2. $\overrightarrow{\text{card}}_{\vec{G}}\text{-ghw}(\vec{G}) \leq \text{card-ghw}(\underline{\vec{G}})$, and

$$3. \overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G}) \leq \text{card-hw}(\vec{G}).$$

Proof. 1,2: Here we can use the fact that $f\text{-cw}(\vec{G}) = f^{\text{mon}}\text{-cw}(\vec{G})$ and $f\text{-ghw}(\vec{G}) = f^{\text{mon}}\text{-ghw}(\vec{G})$ for all width functions f . Clearly $\overrightarrow{\text{card}}_{\vec{G}}^{\text{mon}}(X) \leq \text{card}(X)$ for all $X \subseteq V(\vec{G})$.

3: Suppose the cops have a winning strategy for $\text{RC}_{\text{mon}}(\vec{G}, \text{card}, k)$. We may assume that in every move the set of vertices occupied by the cops either increases or decreases. The cops can clearly win $\text{RC}(\vec{G}, \overrightarrow{\text{card}}_{\vec{G}}, k)$ by playing according to this strategy, except that they occupy $\langle X \rangle$ rather than X , so it only remains to show that this strategy is monotone. Suppose the cops would move from X to X' according to the original strategy. Then the cops actually move from $\langle X \rangle$ to $\langle X' \rangle$. If $X \subseteq X'$, then $\langle X \rangle \subseteq \langle X' \rangle$ and the move is certainly monotone. If $X \supseteq X'$ and the robber's position is r , then let R be the robber's escape space with respect to X or, equivalently, with respect to X' . It is not hard to see that $R \cap \langle X \rangle = R \cap \langle X' \rangle$. From this it easily follows that the move is monotone. $\square$

Let $\vec{G}$ be a mixed graph, and let (T, B, C) be a (generalised) hypertree decomposition of $\vec{G}$ witnessing that $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G}) \leq k$. It is easy to see that all nodes $t \in T$ satisfy $C_t = \langle C_t \rangle$.

The next lemma shows that for a finite mixed graph $\vec{G}$ with $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G}) \leq k$ (or $\overrightarrow{\text{card}}_{\vec{G}}\text{-ghw}(\vec{G}) \leq k$) there is always a particularly nice (generalised) hypertree decomposition witnessing this. We will need this in the next section.

Lemma 5.1.8 *Let $k < \omega$ be an integer, let $\vec{G}$ be a finite mixed graph, and let (T, B, C) be a (generalised) hypertree decomposition of $\vec{G}$ satisfying*

$$\overrightarrow{\text{card}}_{\vec{G}}\text{-width}(T, B, C) \leq k.$$

Then $(T, \langle B \rangle, C)$ is also a (generalised) hypertree decomposition of $\vec{G}$ satisfying $\overrightarrow{\text{card}}_{\vec{G}}\text{-width}(T, \langle B \rangle, C) \leq k$.

Proof. By Lemma 5.1.4, $(T, \langle B \rangle)$ is a tree decomposition of $\vec{G}$, so (HD1) is satisfied. Since all $t \in T$ satisfy $\langle C_t \rangle = C_t$, and $B_t \subseteq C_t$, we have $\langle B_t \rangle \subseteq C_t$, and thus (HD2) holds.

In the case that (T, B, C) is a hypertree decomposition, it remains to show that all $t \in T$ satisfy $C_t \cap \langle B \rangle_{T_t} \subseteq \langle B_t \rangle$ (condition (HD3)). Towards a contradiction, suppose there exists a vertex $x \in (C_t \cap \langle B \rangle_{T_t}) \setminus \langle B_t \rangle$. Then there is a node $s \in T_t$ with $x \in \langle B_s \rangle$. Since $x \notin B_t$ and (by (HD3) of (T, B, C)) we have $B_t \supseteq C_t \cap B_{T_t}$, it follows that $x \notin B_{T_t}$. Thus no node t_x with $x \in B_{t_x}$ is in T_t . But then $x \in (\langle B_{t_x} \rangle \cap \langle B_s \rangle) \setminus \langle B_t \rangle$, a contradiction to connectedness (TD3) of $(T, \langle B \rangle)$. $\square$

Lemma 5.1.9

The width function $\overrightarrow{\text{card}}_{\vec{G}}^{\text{mon}}$ on a finite mixed graph $\vec{G}$ (i. e. on its underlying graph $\underline{\vec{G}}$) is weakly submodular.

Proof. By Remark 2.4.3 it suffices to show that $\overrightarrow{\text{card}}_{\vec{G}}$ is weakly submodular: Let X_1, X_2 be finite subsets of $V(\vec{G})$. If

$$\overrightarrow{\text{card}}_{\vec{G}}(X_1) + \overrightarrow{\text{card}}_{\vec{G}}(X_2) = \infty,$$

then there is nothing to show. Otherwise, there exist $V_1, V_2 \subseteq V(\vec{G})$ such that $\langle V_i \rangle = X_i$ and $|V_i| = \overrightarrow{\text{card}}_{\vec{G}}(X_i)$ for $i \in \{1, 2\}$. Then $\langle V_1 \cup V_2 \rangle = X_1 \cup X_2$ and thus $\overrightarrow{\text{card}}_{\vec{G}}(X_1 \cup X_2) \leq \overrightarrow{\text{card}}_{\vec{G}}(X_1) + \overrightarrow{\text{card}}_{\vec{G}}(X_2)$. $\square$

Corollary 5.1.10 *For a finite mixed graph $\vec{G}$, the width function $\overrightarrow{\text{card}}_{\vec{G}}^{\text{mon}}$ on G satisfies Theorem 2.4.23, 1 and 2.*

In the next section we show that even more inequalities hold. This is done by a different method, since—as we will now see—the width function $\overrightarrow{\text{card}}_{\vec{G}}^{\text{mon}}$ is not additive in general.

Example 5.1.11 *Let $\vec{G}$ be the following mixed graph:*

Mixed graph $\vec{G}$	
vertices:	1, 2, 3
edges:	none
arcs:	(1, 2), (2, 1), (2, 3), (3, 2)

Then the width function $\overrightarrow{\text{card}}_{\vec{G}}^{\text{mon}}$ on $\vec{G}$ is not additive: The sets $\{1\}, \{3\}$ do not touch in $\vec{G}$, but we have

$$\overrightarrow{\text{card}}_{\vec{G}}^{\text{mon}}(\{1, 3\}) = 1 < 2 = \overrightarrow{\text{card}}_{\vec{G}}^{\text{mon}}(\{1\}) + \overrightarrow{\text{card}}_{\vec{G}}^{\text{mon}}(\{3\}).$$

5.1.2 Translation between hypergraphs and mixed graphs

In this section, we show that a finite mixed graph can be implemented as a finite hypergraph and vice versa. As a corollary we obtain linear coherence of $\overrightarrow{\text{card}}_{\vec{G}}\text{-cw}$ and $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}$ on mixed graphs. As another corollary we get that the (open) recognition problem GHW^k and the problem deciding, given a mixed graph $\vec{G}$, whether $\overrightarrow{\text{card}}_{\vec{G}}\text{-ghw}(\vec{G}) \leq k$, are PTIME equivalent.

Implementing arcs as hyperedges

Theorem 5.1.12 *For a finite mixed graph $\vec{G}$ consider the following hypergraph pair $(G_{\vec{G}}, H_{\vec{G}})$.*

Hypergraph pair $(G_{\vec{G}}, H_{\vec{G}})$	
vertex set:	$V(\vec{G})$
edges of $G_{\vec{G}}$:	e , where $e \in E(\vec{G})$ $\langle v \rangle$, where $v \in V(\vec{G})$
edges of $H_{\vec{G}}$:	$\langle v \rangle$, where $v \in V(\vec{G})$

Then

1. $\overrightarrow{\text{card}}_{\vec{G}}\text{-cw}(\vec{G}) = c_{H_{\vec{G}}}\text{-cw}(G_{\vec{G}})$.
2. $\overrightarrow{\text{card}}_{\vec{G}}\text{-ghw}(\vec{G}) = c_{H_{\vec{G}}}\text{-ghw}(G_{\vec{G}})$.
3. $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G}) = c_{H_{\vec{G}}}\text{-hw}(G_{\vec{G}})$.

Proof. 1: Note that $\vec{G} \subseteq G_{\vec{G}}$. Thus, it is easier to catch the robber on $\vec{G}$; and a cop winning strategy for $\text{RC}(G_{\vec{G}}, c_{H_{\vec{G}}}, k)$ is already a winning strategy for $\text{RC}(\vec{G}, \overrightarrow{\text{card}}_{\vec{G}}, k)$.

Conversely, suppose the cops can win $\text{RC}(\vec{G}, \overrightarrow{\text{card}}_{\vec{G}}, k)$. If we show that edges in $E(G_{\vec{G}}) \setminus E(\vec{G})$ do not give an advantage to the robber, then the winning strategy

for $\text{RC}(\vec{G}, \overrightarrow{\text{card}}_{\vec{G}}, k)$ is actually a winning strategy for $\text{RC}(G_{\vec{G}}, c_{H_{\vec{G}}}, k)$. For $\{u, v\} \in E(\underline{G_{\vec{G}}}) \setminus E(\vec{G})$ let

$$S_{\{u,v\}} := \{x \in V(\vec{G}) \mid \{u, v\} \subseteq \langle x \rangle\} \neq \emptyset.$$

Note that for all $x \in S_{\{u,v\}}$ there are paths in $S_{\{u,v\}}$ connecting v to x and u to x . Now suppose the cops fly from $X \subseteq V(\vec{G}) = V(G_{\vec{G}})$ to X' and suppose the robber can use the edge $\{u, v\}$ during the flight. Then $S_{\{u,v\}} \cap (X \cap X') = \emptyset$. But this means that u and v are connected by some $x \in S_{\{u,v\}}$, and thus the robber can also reach u from v and vice versa on $\underline{G_{\vec{G}}}$. Therefore, the robber's escape spaces in $G_{\vec{G}}$ are exactly the robbers escape spaces in $\vec{G}$, and the cops win.

2: Is proved like 3.

3: Since $\underline{G_{\vec{G}}} \subseteq G_{\vec{G}}$, a hypertree decomposition of $G_{\vec{G}}$ witnessing that $c_{H_{\vec{G}}}$ -hw($G_{\vec{G}}$) $\leq k$ already witnesses $\overrightarrow{\text{card}}_{\vec{G}}$ -hw($\vec{G}$) $\leq k$: For $t \in T$ let $Y \subseteq E(H_{\vec{G}})$ witness that $c_{H_{\vec{G}}}(C_t) \leq k$, i. e. $|Y| \leq k$ and

$$C_t = \bigcup Y = \bigcup_{\langle v \rangle \in Y} \langle v \rangle = \langle \{v \mid \langle v \rangle \in Y\} \rangle,$$

where the last equality holds because of Remark 5.1.3. Thus, $\{v \mid \langle v \rangle \in Y\}$ witnesses that $\overrightarrow{\text{card}}_{\vec{G}}(C_t) \leq k$.

Conversely, let (T, B, C) be a hypertree decomposition of $\underline{G_{\vec{G}}}$ witnessing that $\overrightarrow{\text{card}}_{\vec{G}}$ -hw($\vec{G}$) $\leq k$. By Lemma 5.1.8 we may assume that $B = \langle B \rangle$. We show that (T, B, C) is a hypertree decomposition of $G_{\vec{G}}$ witnessing $c_{H_{\vec{G}}}$ -hw($G_{\vec{G}})$ $\leq k$.

(HD1): (T, B) is a tree decomposition of $G_{\vec{G}}$: Since (T, B) is a tree decomposition of $\underline{G_{\vec{G}}}$, every vertex $v \in V(G_{\vec{G}}) = V(\vec{G})$ is covered by some node $t \in T$. Now let $e \in E(G_{\vec{G}})$. If $e \in E(\vec{G})$, then e is covered in some piece of (T, B) by assumption. If $e = \langle v \rangle$ for some $v \in V$, then v is covered in some piece B_{t_v} of T and thus $\langle v \rangle \subseteq B_{t_v} = \langle B_{t_v} \rangle$. Obviously, (T, B) satisfies connectedness (TD3).

(HD2) and (HD3) are satisfied by assumption.

For $t \in T$, let $X \subseteq V(\vec{G})$ witness that $\overrightarrow{\text{card}}_{\vec{G}}(C_t) \leq k$, i. e. $|X| \leq k$ and

$$C_t = \langle X \rangle = \bigcup \{\langle x \rangle \mid x \in X\},$$

where the last equality holds because of Remark 5.1.3. Thus, $\{\langle v \rangle \mid v \in X\}$ witnesses that $c_{H_{\vec{G}}}(C_t) \leq k$. Altogether, (T, B, C) is a hypertree decomposition of $G_{\vec{G}}$ witnessing $c_{H_{\vec{G}}}$ -hw($G_{\vec{G}})$ $\leq k$. $\square$

Corollary 5.1.13 *Every finite mixed graph $\vec{G}$ satisfies*

$$\begin{aligned} \overrightarrow{\text{card}}_{\vec{G}}\text{-cw}(\vec{G}) &\leq \overrightarrow{\text{card}}_{\vec{G}}\text{-ghw}(\vec{G}) \leq \overrightarrow{\text{card}}_{\vec{G}}\text{-cw}_{\text{mon}}(\vec{G}) = \overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G}) \\ &\leq 3 \cdot \overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G}) + 1. \end{aligned}$$

Proof. The hypergraph pair $(G_{\vec{G}}, H_{\vec{G}})$ from Theorem 5.1.12 is obviously trivial, so we can apply Corollary 3.2.9. $\square$

Implementing hyperedges as arcs

Theorem 5.1.14 *For a finite hypergraph H consider the following directed graph $\vec{G}_H$.*

Directed graph $\vec{G}_H$	
vertex set:	$V(H) \dot{\cup} E(H)$
arcs:	(e, v) , where $v \in e \in E(H)$

Then

1. $\overrightarrow{\text{card}}_{\vec{G}}\text{-cw}(\vec{G}_H) = c_H\text{-cw}(H)$
2. $\overrightarrow{\text{card}}_{\vec{G}}\text{-ghw}(\vec{G}_H) = c_H\text{-ghw}(H)$
3. $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G}_H) = c_H\text{-hw}(H)$

Proof. Before beginning with the proof, we observe that since H is nonempty, using Corollary 2.2.20 we have

$$1 \leq c_H\text{-cw}(H) \leq c_H\text{-ghw}(H) \leq c_H\text{-hw}(H).$$

3: First we show $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G}_H) \leq c_H\text{-hw}(H)$. Let (T, B, C) be a hypertree decomposition of $\underline{H}$ of c_H -width $\leq k$, for some integer $k \geq 0$. We may assume that for each $e \in E(H)$ there exists a leaf $t_e \in T$ with $B_{t_e} = C_{t_e} = e$. (Otherwise choose a node $t \in T$ with $e \subseteq B_t$ and add such a leaf t_e to T , connecting it with t .) For $t \in T$ define

$$B'_t := \begin{cases} B_t & \text{if } t \notin \{t_e \mid e \in E(H)\}, \\ B_t \cup \{e\} & \text{if } t = t_e \text{ for some edge } e \in E(H). \end{cases}$$

and

$$C'_t = \begin{cases} C_t & \text{if } t \notin \{t_e \mid e \in E(H)\}, \\ B_t \cup \{e\} & \text{if } t = t_e \text{ for some edge } e \in E(H). \end{cases}$$

We now show that (T, B', C') is a hypertree decomposition of $\vec{G}_H$ that witnesses $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G}_H) \leq k$:

(HD1): (T, B') is a tree decomposition for $\vec{G}_H$: Every $v \in V(\vec{G}_H) \cap V(H)$ is covered by some $B_t \subseteq B'_{T_t}$. Every $e \in V(\vec{G}_H) \cap E(H)$ is covered in B'_{t_e} . Moreover, every edge $\{e, v\} \in E(\vec{G}_H)$ satisfies $\{e, v\} \subseteq B_{t_e}$, and connectedness obviously holds as well.

(HD2): Obviously, each $t \in T$ satisfies $B'_t \subseteq C'_t$.

(HD3): We have to show that all $t \in T$ satisfy $C'_t \cap B'_{T_t} \subseteq B_t$.

For $t \in T \setminus \{t_e \mid e \in E(H)\}$ we have

$$C'_t \cap B'_{T_t} = C_t \cap (B_{T_t} \cup \{e \mid e \in E(H), t_e \in T_t\}) = C_t \cap B_{T_t} \subseteq B_t = B'_t.$$

If $t = t_e$ for some $e \in E(H)$, then t is a leaf and $C'_t \cap B'_{T_t} = B'_t$. This proves (HD3).

Let $t \in T \setminus \{t_e \mid e \in E\}$. Choose $Y_t \subseteq E$ with $|Y_t| \leq k$ and $C_t = \bigcup Y_t$. Then $Y_t \subseteq V(\vec{G}_H)$ and $\langle Y_t \rangle = C_t = C'_t$. Thus $\overrightarrow{\text{card}}_{\vec{G}}(C'_t) \leq k$. For $t = t_e$ we have $C'_t = B_t \cup \{e\} = \langle e \rangle$. Thus $\overrightarrow{\text{card}}_{\vec{G}}(C'_t) \leq 1 \leq c_H\text{-hw}(H) \leq k$.

Altogether, (T, B', C') is a hypertree decomposition of $\vec{G}_H$ witnessing $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G}_H) \leq k$.

Conversely, using the game theoretic characterisation, we show

$$\overrightarrow{\text{card}}_{\vec{G}}\text{-cw}_{\text{mon}}(\vec{G}_H) \geq c_H\text{-cw}_{\text{mon}}(H).$$

Suppose the cops have a winning strategy for $\text{RC}_{\text{mon}}(\vec{G}_H, \overrightarrow{\text{card}}_{\vec{G}}, k)$. Choose a function $\varphi : V(H) \rightarrow E(H)$ satisfying $v \in \varphi(v)$. For a set $X \subseteq V(\vec{G}_H)$, define the set $\varphi(X) = (X \cap E(H)) \cup \{\varphi(v) \mid v \in X \cap V(H)\}$.

The cops play on $\underline{H}$ according to their winning strategy on $\vec{G}_H$, placing the cops on $\varphi(X) \subseteq E(H)$ instead of $X \subseteq V(H) \cup E(H) = V(\vec{G}_H)$. Then a position (X, R) of $\text{RC}_{\text{mon}}(\vec{G}_H, \overrightarrow{\text{card}}_{\vec{G}}, k)$ yields a position $(\bigcup \varphi(X), R')$ on H where $R' \subseteq R \cap V(H)$, so the cops can win. It remains to show that their strategy is monotone.

Suppose towards a contradiction, that while the move from (X, R) to (Y, S) is monotone, the move from $(\bigcup \varphi(X), R')$ to $(\bigcup \varphi(Y), S')$ is not. Then there exists a vertex $v \in S' \setminus R'$. Let $u \in R'$ be the last vertex in R' on a path from R' to v in $\underline{H}$ and let u' be the next vertex after u on the path to v . Then (because $u' \in \bigcup \varphi(X)$) we have $\{e \in E(H) \mid u' \in e\} \cap \varphi(X) \neq \emptyset$ but $\{e \in E(H) \mid u' \in e\} \cap \varphi(Y) = \emptyset$ (because $u' \notin \bigcup \varphi(Y)$). Thus for

$$M := \{v \in V(H) \mid u' \in \varphi(v)\} \cup \{e \in E(H) \mid u' \in e\} \subseteq V(\vec{G}_H)$$

we have $M \cap X \neq \emptyset$ but $M \cap Y = \emptyset$. Thus we have $\{v \in V(H) \mid u' \in \varphi(v)\} \cap X \neq \emptyset$ or $\{e \in E(H) \mid u' \in e\} \cap X \neq \emptyset$.

First assume that there exists a vertex

$$v_0 \in \{v \in V \mid u' \in \varphi(v)\} \cap X.$$

Then $v_0 \notin Y$. Since $R' \subseteq R \cap V(H)$ we have $u \in R$. Since u and u' are neighbours in $\underline{H}$, there exists an edge $e \in E(H)$ with $\{u, u'\} \subseteq e$. This edge e must be free during the flight from X to Y , and thus in $\vec{G}_H$ there is a path from u via e to u' . Since during the cops' flight from X to Y the set M is free, the edge $\varphi(v_0)$ as a vertex of $\vec{G}_H$ is free. By the choice of v_0 , we have $u' \in \varphi(v_0)$ and thus there is a path from u' via $\varphi(v_0)$ to v_0 , a contradiction to monotonicity of the strategy for $\text{RC}_{\text{mon}}(\vec{G}_H, \overrightarrow{\text{card}}_{\vec{G}}, k)$.

Second, assume that there exists an edge $e_0 \in \{e \in E(H) \mid u' \in e\} \cap X$. Then e_0 is free during the flight from X to Y and the robber can reach e_0 via u' , a contradiction.

Thus we have found a strategy for $\text{RC}_{\text{mon}}(\underline{H}, c_H, k)$.

2: $\overrightarrow{\text{card}}_{\vec{G}}\text{-ghw}(\vec{G}_H) \leq c_H\text{-ghw}(H)$ is proved like $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G}_H) \leq c_H\text{-hw}(H)$ (see 3). For $\overrightarrow{\text{card}}_{\vec{G}}\text{-ghw}(\vec{G}_H) \geq c_H\text{-ghw}(H)$ use the fact that a width function f of a graph G satisfies $f\text{-ghw}(G) = f^{\text{mon}}\text{-hw}(G) = f^{\text{mon}}\text{-cw}_{\text{mon}}(G)$ and show

$$\overrightarrow{\text{card}}_{\vec{G}}^{\text{mon}}\text{-cw}_{\text{mon}}(\vec{G}_H) \geq c_H^{\text{mon}}\text{-cw}_{\text{mon}}(H).$$

The proof is analogous to the proof of $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G}_H) \geq c_H\text{-hw}(H)$ (see 3).

1: $\overrightarrow{\text{card}}_{\vec{G}}\text{-cw}(\vec{G}_H) \geq c_H\text{-cw}(H)$ follows from the proof of $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G}_H) \geq c_H\text{-hw}(H)$ (see 3). For $\overrightarrow{\text{card}}_{\vec{G}}\text{-cw}(\vec{G}_H) \leq c_H\text{-cw}(H)$, assume that the cops have a winning strategy for $\text{RC}(\underline{H}, c_H, k)$. Note that although the robber has more vertices and edges in $\vec{G}_H$ than in $\underline{H}$, there are no new paths between vertices $\{u, v\} \subseteq V(H)$. Thus if (X, R) is a position of $\text{RC}(\underline{H}, c_H, k)$, then (X, R') where $R' = R \cap V(H)$ is a position of the game on $\vec{G}_H$. Now the cops win as follows: As long as the robber's escape space contains a vertex of $V(H)$, the cops play according to this strategy on $\vec{G}_H$. If the robber's escape space does not contain a vertex of $V(H)$, then it is $\{e\}$ for some $e \in E(H)$. Then the robber is caught by one cop moving to e . $\square$

Recall that GHW^k is the problem of deciding for a given hypergraph H whether $c_H\text{-ghw}(H) \leq k$.

Corollary 5.1.15 *Let $k \in \omega$. The problem GHW^k and the problem of deciding, given a mixed graph $\vec{G}$, whether $\overrightarrow{\text{card}}_{\vec{G}}\text{-ghw}(\vec{G}) \leq k$, are PTIME equivalent.*

Proof. Given a hypergraph H , we can compute $\vec{G}_H$ of Theorem 5.1.14 in polynomial time. Conversely, given a mixed graph $\vec{G}$, we compute $(G_{\vec{G}}, H_{\vec{G}})$ from Theorem 5.1.12 in polynomial time. By Corollary 5.1.13 the hypergraph pair $(G_{\vec{G}}, H_{\vec{G}})$ is trivial. Therefore we can use Theorem 3.2.8: If $H_{\vec{G}}$ is not tame, then reject. Otherwise, $n := |E(H_{\vec{G}})| + 1$ satisfies $n > c_{H_{\vec{G}}}\text{-hw}(G_{\vec{G}})$. With this n we can even compute the hypergraph $J_{(G_{\vec{G}}, H_{\vec{G}})}$ of Theorem 3.2.8 in polynomial time. This completes the proof, because

$$\overrightarrow{\text{card}}_{\vec{G}}\text{-ghw}(\vec{G}) = c_{H_{\vec{G}}} - \text{ghw}(G_{\vec{G}}) = c_{J_{(G_{\vec{G}}, H_{\vec{G}})}} - \text{ghw}(J_{(G_{\vec{G}}, H_{\vec{G}})}). \quad \square$$

5.2 Directed hypergraphs

Directed hypergraphs are more general than mixed graphs. While every structure has a natural underlying directed hypergraph (see Section 5.2.2), only those structures where all partial functions are unary have an underlying mixed graph. Moreover, directed hypergraphs $\vec{H}$ are a natural generalisation of hypergraphs if we want to define a width function $\overrightarrow{\text{card}}_{\vec{H}}$ similar to $\overrightarrow{\text{card}}_{\vec{G}}$.

Our function $\overrightarrow{\text{card}}_{\vec{H}}$ defined below on directed hypergraphs seems to be more general than $\overrightarrow{\text{card}}_{\vec{G}}$: Only in the case that all partial functions are unary, we were able to find an implementation similar to the implementation of mixed graphs as hypergraphs in the last section (see Theorem 5.2.14 below).

Finally, we show that for a fixed integer $k > 0$, the mixed graphs $\vec{G}$ with $\overrightarrow{\text{card}}_{\vec{G}}\text{-hw}(\vec{G}) \leq k$ and the directed hypergraphs $\vec{H}$ with $\overrightarrow{\text{card}}_{\vec{H}}\text{-hw}(\vec{H}) \leq k$ are recognisable in polynomial time. Moreover, we show that the problem $\text{HOM}(\mathcal{C}, _)$ is solvable in polynomial time for a class $\mathcal{C}$ of structures such that the underlying directed hypergraph $\vec{H}$ of the core of each structure from $\mathcal{C}$ satisfies $\overrightarrow{\text{card}}_{\vec{H}}\text{-ghw}(\vec{H}) \leq k$ (for fixed k).

5.2.1 Definitions and some observations

Definition 5.2.1 A directed hypergraph is a pair $\vec{H} = (V(\vec{H}), D(\vec{H}))$ consisting of a non-empty set $V(\vec{H})$ of vertices and a set $D(\vec{H}) \subseteq \mathcal{P}_{<\omega}(V(\vec{H})) \times \mathcal{P}_{<\omega}(V(\vec{H}))$ of pairs of finite sets of vertices, called the hyperarcs of $\vec{H}$.

If $(h_1, h_2) \in D(\vec{H})$ is a hyperarc of $\vec{H}$, then h_1 is called the source set and h_2 is called the target set. We will usually write $h_1 \rightarrow h_2$ for such a hyperarc. In the special case where the target set $h_2 = \emptyset$ is empty, we will just write h_1 instead of $h_1 \rightarrow \emptyset$. We call such hyperarcs hyperedges.

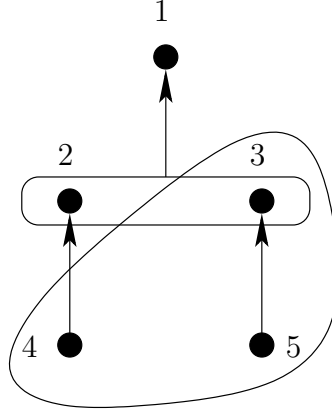
We identify hypergraphs (V, E) with the directed hypergraphs $(V, E \times \{\emptyset\})$. We also identify mixed graphs (V, E, D) with the directed hypergraphs $(V, E \times \{\emptyset\} \cup \{(\{u\}, \{v\}) \mid (u, v) \in D\})$.

The underlying hypergraph $\underline{H}$ of a directed hypergraph $\vec{H}$ is defined as follows.

Hypergraph $\underline{H}$	
vertex set:	$V(\vec{H})$
edges:	$h_1 \cup h_2$, where $(h_1, h_2) \in D(\vec{H})$

Example 5.2.2 Figure 5.2 shows the following directed hypergraph $\vec{H}$.

Directed hypergraph $\vec{H}$	
vertices:	1, 2, 3, 4, 5
hyperarcs:	$\{3, 4, 5\}$
	$\{2, 3\} \rightarrow \{1\}, \{4\} \rightarrow \{2\}, \{5\} \rightarrow \{3\}$

Figure 5.2: The directed hypergraph $\vec{H}$ of Example 5.2.2.

It appears that the terminology around directed hypergraphs is not completely standardised yet. The most common definition of directed hypergraphs is a bit more restrictive than the one given above: For hyperarcs $h_1 \rightarrow h_2$, the set h_2 must consist of precisely one element. This is not appropriate in our context, where we also need a way to encode undirected hyperedges.

For our purposes it would be sufficient to permit only hyperarcs $h_1 \rightarrow h_2$ where $h_2 \in \mathcal{P}_{\leq 1}(V(\vec{H}))$ is either empty or consists of a single vertex. This is in fact a natural restriction for one of our applications (underlying directed hypergraphs of structures with function symbols), but not for the other (underlying directed hypergraphs of databases or queries, in the presence of functional dependencies).

Definition 5.2.3 Let $\vec{H}$ be a directed hypergraph. The forward closure of $X \subseteq V$ is the smallest superset $\langle X \rangle \subseteq V(\vec{H})$ of X which is closed under out-hyperarcs, i. e. if $(h_1, h_2) \in D(\vec{H})$ is a hyperarc such that $h_1 \subseteq \langle X \rangle$, then $h_2 \subseteq \langle X \rangle$.

Remark 5.2.4 For a directed hypergraph $\vec{H}$, let $X, Y \subseteq V(\vec{H})$. Then

1. $\langle X \rangle \cup \langle Y \rangle \subseteq \langle X \cup Y \rangle$.
2. If $D(\vec{H}) \subseteq \mathcal{P}_{\leq 1}(V(\vec{H})) \times \mathcal{P}_{< \omega}(V(\vec{H}))$, then $\langle X \rangle \cup \langle Y \rangle = \langle X \cup Y \rangle$. □

Of course, for directed hypergraphs, $\langle X \rangle \cup \langle Y \rangle = \langle X \cup Y \rangle$ is not true in general:

Example 5.2.5 Let $\vec{H}$ be the directed hypergraph of Example 5.2.2 (Figure 5.2), and let $X := \{4, 5\} \subseteq V(\vec{H})$. Then

$$\bigcup_{v \in X} \langle v \rangle = \langle 4 \rangle \cup \langle 5 \rangle = \{2, 4\} \cup \{3, 5\} \subsetneq \{1, 2, 3, 4, 5\} = \langle X \rangle.$$

It is not hard to find examples showing that the difference of the sizes $|\bigcup_{v \in X} \langle v \rangle|$ and $|\langle X \rangle|$ can actually be unbounded. We will do this below (in Proposition 5.2.13).

The following lemma is proved like Lemma 5.1.4.

Lemma 5.2.6 Let $\vec{H}$ be a finite directed hypergraph and let (T, B) be a tree decomposition of its underlying hypergraph $\underline{H}$. Then $(T, \langle B \rangle)$ is also a tree-decomposition of $\vec{H}$. □

By a *width function* on a directed hypergraph $\vec{H}$ we understand a width function on its underlying hypergraph $\underline{H}$.

Definition 5.2.7 For finite directed hypergraphs $\vec{H}$ we consider the following width function $\vec{c}_{\vec{H}}$.

$$\begin{aligned} \vec{c}_{\vec{H}} : \mathcal{P}_{<\omega}(V(\vec{H})) &\rightarrow \mathbb{R} \cup \{\infty\} \\ X &\mapsto \inf \{ |Y| \mid Y \subseteq E(\vec{H}), X = \langle \bigcup Y \rangle \}. \end{aligned}$$

As for mixed graphs, $\vec{c}_{\vec{H}}$ is not monotone, and it does not make much sense to define $\vec{c}_{\vec{H}}$ for arbitrary infinite graphs. We write $\vec{c}_{\vec{H}}\text{-hw}(\vec{H})$ for $\vec{c}_{\vec{H}}\text{-hw}(\vec{\underline{H}})$; analogously for the other $\vec{c}_{\vec{H}}$ -invariants.

Example 5.2.8

- The directed hypergraph $\vec{H}$ of Example 5.2.2 (Figure 5.2) satisfies $\vec{c}_{\vec{H}}\text{-hw}(\vec{H}) = 1$.
- If $\vec{H}$ is a finite hypergraph (i. e. $D(\vec{H}) \subseteq \mathcal{P}_{<\omega} \times \{\emptyset\}$), then

$$\vec{c}_{\vec{H}}\text{-hw}(\vec{H}) = c_{\underline{H}}\text{-hw}(\underline{H}).$$

The following example shows that $c_{\underline{H}}\text{-hw}$ and $\vec{c}_{\vec{H}}\text{-hw}$ are not coherent.

Example 5.2.9 For every even integer $n > 0$ let $\vec{K}_n$ be the mixed graph from Example 5.1.6, regarded as a directed hypergraph. Then

- $\vec{c}_{\vec{K}_n}\text{-hw}(\vec{K}_n) = 1$,
- $c_{\underline{K}_n}\text{-hw}(\underline{K}_n) = \frac{n}{2}$.

Lemma 5.2.10 Let $\vec{H}$ be a finite directed hypergraph. Then

- $\vec{c}_{\vec{H}}\text{-cw}(\vec{H}) \leq c_{\underline{H}}\text{-cw}(\underline{H})$,
- $\vec{c}_{\vec{H}}\text{-ghw}(\vec{H}) \leq c_{\underline{H}}\text{-ghw}(\underline{H})$, and
- $\vec{c}_{\vec{H}}\text{-hw}(\vec{H}) \leq c_{\underline{H}}\text{-hw}(\underline{H})$.

Proof. This is proved precisely in the same way as Lemma 5.1.7. $\square$

Lemma 5.2.11 Let $k < \omega$ be an integer and $\vec{H}$ a finite directed hypergraph with $\vec{c}_{\vec{H}}\text{-hw}(\vec{H}) \leq k$, witnessed by a hypertree decomposition (T, B, C) . Then $(T, \langle B \rangle, C)$ also witnesses $\vec{c}_{\vec{H}}\text{-hw}(\vec{H}) \leq k$.

Proof. We show that $(T, \langle B \rangle, C)$ is also a hypertree decomposition witnessing $\vec{c}_{\vec{H}}\text{-hw}(\vec{H}) \leq k$. (HD1): By Lemma 5.2.6, $(T, \langle B \rangle)$ is a tree decomposition of $\underline{H}$.

The proofs of (HD2) and (HD3) are literally as in the proof of Lemma 5.1.8. $\square$

Example 5.2.12 For every integer $n \geq 1$ we consider the following directed hypergraph $\vec{H}_n$ (see Figure 5.3).

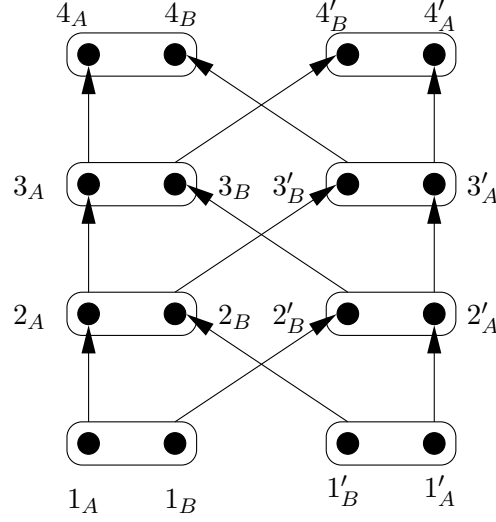


Figure 5.3: The directed hypergraph $\vec{H}_n$ of Example 5.2.12 for $n = 4$. Any two vertices are connected by an undirected edge not depicted here.

Directed hypergraph $\vec{H}_n$

vertices:	$1_A, \dots, n_A, 1_B, \dots, n_B, 1'_B, \dots, n'_B, 1'_A, \dots, n'_A$
hyperarcs:	$\{1_A, 1_B\} \rightarrow \{2_A\}, \dots, \{(n-1)_A, (n-1)_B\} \rightarrow \{n_A\}$ $\{1_A, 1_B\} \rightarrow \{2'_B\}, \dots, \{(n-1)_A, (n-1)_B\} \rightarrow \{n'_A\}$ $\{1'_B, 1'_A\} \rightarrow \{2_B\}, \dots, \{(n-1)'_B, (n-1)'_A\} \rightarrow \{n_B\}$ $\{1'_B, 1'_A\} \rightarrow \{2'_A\}, \dots, \{(n-1)'_B, (n-1)'_A\} \rightarrow \{n'_A\}$ $\{u, v\}, \text{ where } u \neq v$

Proposition 5.2.13 *For every integer $n \geq 1$ the directed hypergraph $\vec{H}_n$ from Example 5.2.12 satisfies*

1. $\vec{c}_{\vec{H}_n}(V(\vec{H}_n)) = \vec{c}_{\vec{H}_n}\text{-ghw}(\vec{H}_n) = 2$,
2. $\min \{ |Y| \mid Y \subseteq E(\vec{H}_n), V(\vec{H}_n) = \bigcup_{h \in Y} \langle h \rangle \} \geq n$, and
3. $c_{\vec{H}_n}(V_n) = c_{\vec{H}_n}\text{-ghw}(\vec{H}_n) > n$.

Thus, by 1 and 3, $\vec{c}_{\vec{H}}$ -ghw and $c_{\vec{H}}$ -ghw are not coherent. From 1 and 2 it follows that there is a set $Y_n \subseteq E(\vec{H}_n)$ showing that the difference of the sizes of $|\bigcup Y_n|$ and $|\bigcup_{h \in Y_n} \langle h \rangle|$ is unbounded.

Proof of Lemma 5.2.13. 1: Since $\langle \{1_A, 1_B\} \cup \{1'_A, 1'_B\} \rangle = V(\vec{H}_n)$, it follows that $\vec{c}_{\vec{H}_n}(V(\vec{H}_n)) \leq 2$. It is easy to see that this is the smallest we can get, and thus $\vec{c}_{\vec{H}_n}(V(\vec{H}_n)) = 2$. The second equality follows from this, together with the fact that $\vec{H}_n$, the underlying graph of the underlying hypergraph of $\vec{H}_n$, is a clique and hence every tree decomposition of $\vec{H}_n$ has a piece $V(\vec{H}_n)$.

2: For every $h \in E(\vec{H}_n)$ we have $|\langle h \rangle| \leq 4$. Since $|V_n| = 4n$, we have

$$\min \{ |Y| \mid Y \subseteq E(\vec{H}_n), V(\vec{H}_n) = \bigcup_{h \in Y} \langle h \rangle \} \geq \frac{4n}{4} = n.$$

3: For every $h \in E(\vec{H}_n)$ we have $|h| \leq 3$. Thus, for $c_{\vec{H}_n}(V(\vec{H}_n)) > n$ we can argue as in 2. The rest follows from this, together with the fact that $\vec{H}_n$ is a clique and hence every tree decomposition of $\vec{H}_n$ has a piece $V(\vec{H}_n)$. $\square$

Theorem 5.2.14 *For a finite directed hypergraph $\vec{H}$ consider the following hypergraph H .*

Hypergraph H	
vertex set:	$V(\vec{H})$
edges:	$\langle e \rangle$, where $e \in E(\vec{H})$

Then

1. $\vec{c}_{\vec{H}}\text{-cw}(\vec{H}) \leq c_H\text{-cw}(H)$,
2. $\vec{c}_{\vec{H}}\text{-ghw}(\vec{H}) \leq c_H\text{-ghw}(H)$,

If $\vec{H}$ is actually a mixed graph or, more generally, $D(\vec{H}) \subseteq \mathcal{P}_1(V(\vec{H})) \times \mathcal{P}_{<\omega}(V(\vec{H}))$, then

1. $\vec{c}_{\vec{H}}\text{-cw}(\vec{H}) = c_H\text{-cw}(H)$,
2. $\vec{c}_{\vec{H}}\text{-ghw}(\vec{H}) = c_H\text{-ghw}(H)$,
3. $\vec{c}_{\vec{H}}\text{-hw}(\vec{H}) = c_H\text{-hw}(H)$.

Proof. For the first inequality, observe that $\vec{H} \subseteq H$. A cop winning strategy for $\text{RC}(H, c_H, k)$ yields a winning strategy for $\text{RC}(\vec{H}, \vec{c}_{\vec{H}}, k)$ as follows: Whenever the cops move to a set $\bigcup Y \subseteq V(H)$ on H with $|Y| \leq k$, they move to $\langle \bigcup_{e \in Y} e \rangle \subseteq V(\vec{H})$ on $\vec{H}$. By Remark 5.2.4, 1, we have $\bigcup Y \subseteq \langle \bigcup_{e \in Y} e \rangle$, and thus the cops win.

For the second inequality, let (T, B, C) be a generalised hypertree decomposition witnessing that $c_H\text{-ghw}(H) \leq k$. Since $\vec{H} \subseteq H$, (T, B) is a tree decomposition of $\vec{H}$. For every node $t \in T$ choose a set $Y_t \subseteq E(H)$ with $|Y_t| \leq k$ and $\bigcup Y_t = C_t$. Define $C'_t = \langle \bigcup_{e \in Y_t} e \rangle$. Then $\vec{c}_{\vec{H}}(C'_t) \leq k$. Using Remark 5.2.4, 1, we have $B_t \subseteq \bigcup Y_t \subseteq \langle \bigcup_{e \in Y_t} e \rangle$. Thus (T, B, C') is a generalised hypertree decomposition of

The last three equalities are proved similarly to Theorem 5.1.12, where we use Lemma 5.2.11 instead of Lemma 5.1.8, and Remark 5.2.4, 2, instead of Remark 5.1.3. $\square$

Corollary 5.2.15 *Let $\mathcal{C}$ be the class of all directed hypergraph $\vec{H}$ satisfying $D(\vec{H}) \subseteq \mathcal{P}_1(V(\vec{H})) \times \mathcal{P}_{<\omega}(V(\vec{H}))$. Then there is a polynomial time reduction from the problem deciding, given a directed hypergraph $\vec{H} \in \mathcal{C}$, whether $\vec{c}_{\vec{H}}\text{-ghw}(\vec{H}) \leq k$, to the problem GHW^k .*

Proof. Given $\vec{H} \in \mathcal{C}$, we can compute the hypergraph H from Theorem 5.2.14 in polynomial time. $\square$

For applying Chapter 2 to the width functions $\overrightarrow{\text{card}}_{\vec{G}}$ and $\vec{c}_{\vec{H}}$, we take a look at their properties.

Lemma 5.2.16 *The width function $\vec{c}_{\vec{H}}^{\text{mon}}$ on a finite directed hypergraph $\vec{H}$ (i. e. on its underlying graph $\vec{H}$) is weakly submodular.*

Proof. Let X_1, X_2 be finite subsets of $V(\vec{H})$. If $\bar{c}_{\vec{H}}^{\text{non}}(X_1) + \bar{c}_{\vec{H}}^{\text{non}}(X_2) = \infty$, then there is nothing to show. Otherwise, there exist $E_1, E_2 \subseteq E(\vec{H})$ such that $\langle \bigcup E_i \rangle \supseteq X_i$ and $|E_i| = \bar{c}_{\vec{H}}^{\text{non}}(X_i)$ for $i \in \{1, 2\}$. Then by Remark 5.2.4,

$$\langle \bigcup E_1 \cup \bigcup E_2 \rangle \supseteq \langle \bigcup E_1 \rangle \cup \langle \bigcup E_2 \rangle \supseteq X_1 \cup X_2,$$

and thus $\bar{c}_{\vec{H}}^{\text{non}}(X_1 \cup X_2) \leq \bar{c}_{\vec{H}}^{\text{non}}(X_1) + \bar{c}_{\vec{H}}^{\text{non}}(X_2)$. $\square$

Corollary 5.2.17 *For a directed hypergraph $\vec{H}$, the width function $\bar{c}_{\vec{H}}^{\text{non}}$ on $\vec{H}$ satisfies Theorem 2.4.23, 1 and 2.*

It is easy to see that for the directed hypergraph $\vec{G}$ from Example 5.1.11 (which even is a mixed graph) the width function $\bar{c}_{\vec{G}}^{\text{non}}$ is not additive.

5.2.2 More on the homomorphism problem for non-relational signatures

Definition 5.2.18 *Let σ be a signature that may contain partial function symbols as well as relation symbols, and let $\mathcal{M}$ be a σ -structure. The underlying directed hypergraph $\vec{H}_{\mathcal{M}}$ of $\mathcal{M}$ is defined as follows.*

Directed hypergraph $\vec{H}_{\mathcal{M}}$

vertex set: M

hyperarcs: $\{a_1, \dots, a_n\} \rightarrow \{b\},$
 where $\mathcal{M} \models f(a_1, \dots, a_n) = b$ for a partial function symbol f ,
 $\{a_1, \dots, a_n\},$
 where $\mathcal{M} \models Ra_1 \dots a_n$ for a relation symbol R

If σ contains only unary function symbols, then $\vec{H}_{\mathcal{M}}$ is in fact a mixed graph, and we may call $\vec{H}_{\mathcal{M}}$ the underlying mixed graph of $\mathcal{M}$.

Of course, if σ is relational, then $\vec{H}_{\mathcal{M}} = H_{\mathcal{M}}$ is just the underlying hypergraph of $\mathcal{M}$.

Example 5.2.19

- Consider the signature $\sigma = \{p, q, R\}$ where p and q are unary partial function symbols, and R is a binary relation symbol. Define the σ -structure $\mathcal{M}$ by

$$\begin{aligned} M &:= \{1, 2, 3, 4\}, \\ p^M &:= \{(1, 2), (3, 4)\}, \\ q^M &:= \{(1, 3)\}, \text{ and} \\ R^M &:= \{\{1, 4\}, \{2, 3\}, \{2, 4\}\}. \end{aligned}$$

Then the underlying directed hypergraph $\vec{H}_{\mathcal{M}}$ of $\mathcal{M}$ is the mixed graph depicted in Figure 5.1.

- Consider the signature $\sigma = \{p, q, R\}$ where p is a unary partial function symbol, q is a binary partial function symbol, and R is a ternary relation symbol. Define the σ -structure $\mathcal{M}$ by

$$\begin{aligned} M &:= \{1, 2, 3, 4, 5\}, \\ p^M &:= \{(4, 2), (5, 3)\}, \\ q^M &:= \{((2, 3), 1)\}, \text{ and} \\ R^M &:= \{\{3, 4, 5\}\}. \end{aligned}$$

Then the underlying directed hypergraph $\vec{H}_{\mathcal{M}}$ of $\mathcal{M}$ is the directed hypergraph depicted in Figure 5.2.

Let f be a family of width functions $f^{\vec{H}}$, for all directed hypergraphs $\vec{H}$. For example, $f^{\vec{H}} = \vec{c}_{\vec{H}}$. For $k > 0$ we define $\mathcal{C}_k^f := \{\mathcal{M} \mid f^{\vec{H}_{\mathcal{M}}}(\vec{H}_{\mathcal{M}}) \leq k\}$.

Lemma 5.2.20 *Let $k > 0$ be an integer and let f be the family of width functions $\vec{c}_{\vec{H}}$, for each directed hypergraph $\vec{H}$. Then*

1. $\text{SUBSTR}_{\text{enum}}^{\mathcal{C}_k^f} \in \text{P}$, and
2. $\text{HOM}_{\text{enum}}(\mathcal{C}_k^f, -) \in \text{P}$.

Proof sketch. 1: Let $\mathcal{M}$ be a structure. For every set $X \subseteq E(\vec{H}_{\mathcal{M}})$ such that $|X| \leq k$ compute $(\bigcup X)$. This, as well as the removal of any duplicate substructures in the output, can be achieved in polynomial time.

2: We informally describe how to enumerate the homomorphisms from $\mathcal{M}$ to $\mathcal{N}$. Given $\mathcal{M} \in \mathcal{C}_k^f$ we can find a generating set $X \subseteq E(\vec{H}_{\mathcal{M}})$ such that $|X| \leq k$. Thus we have k tuples, each of which occurs in a relation $R_i^{\mathcal{M}}$ in $\mathcal{M}$. Under a homomorphism every such tuple must be mapped to a tuple that occurs in $R_i^{\mathcal{N}}$. Enumerate all ways of mapping the k tuples in this way. If two of the tuples in $\mathcal{M}$ overlap, form the join. (I. e. remove those mappings which are inconsistent because one element would have to be mapped to two different elements of $\mathcal{N}$). Remove those mappings which are not homomorphisms. It is not hard to see that all this can be achieved in polynomial time. $\square$

The same holds for the family f of width functions $\overrightarrow{\text{card}}_{\vec{G}}$, for each mixed graph $\vec{G}$, if we restrict the problem to structures where all partial functions are unary.

Corollary 5.2.21 *Let $k > 0$ be an integer and let f be the family of width functions $\vec{c}_{\vec{H}}$, for each directed graph $\vec{H}$. Then*

1. $\text{HOM}(\text{COREDECOMPOSABLE}_{\text{ghd}}^{\mathcal{C}_k^f}, -) \in \text{P}$,
2. $\text{HD}^{\mathcal{C}_k^f} \in \text{P}$,
3. $\text{CS}^{\mathcal{C}_k^f} \in \text{P}$

Again, the same holds for the family f of width functions $\overrightarrow{\text{card}}_{\vec{G}}$, for each mixed graph $\vec{G}$, if we restrict ourselves to structures where all partial functions are unary.

Proof. Use the above lemma and Theorems 2.3.13, 2.3.16, and Corollary 2.3.19. $\square$

5.3 Fractional edge covers

In this chapter, we define the width function fc_H for a hypergraph H . The corresponding notion of fc_H -hypertree-width of H is the fractional hypertree-width of H as recently introduced by M. Grohe and D. Marx in [GM05]. While the fractional hypertree-width of a finite hypergraph is always rational, we show that for every real number $r > 0$ there is an infinite hypergraph with fractional hypertree-width r . Applying our general theory we show that the ‘fractional’ versions of the inequalities from Theorem 2.4.23 hold.

5.3.1 Definitions and some observations

Definition 5.3.1 Let H be a hypergraph.

- $\mathbb{R}_{\geq 0}^{(E(H))} = \{\psi : E(H) \rightarrow \mathbb{R}_{\geq 0} \mid \psi \text{ is zero almost everywhere}\}.$
- The weight of $\psi \in \mathbb{R}_{\geq 0}^{(E(H))}$ at $v \in V(H)$ is

$$\text{weight}_v(\psi) = \sum_{\substack{h \ni v \\ h \in E(H)}} \psi(h).$$

- The weight of $\psi \in \mathbb{R}_{\geq 0}^{(E(H))}$ is

$$\text{weight}(\psi) = \sum_{h \in E(H)} \psi(h) \in \mathbb{R}_{\geq 0}.$$

Definition 5.3.2 Let H be a finite hypergraph.

- A fractional edge cover of H is a mapping $\psi \in \mathbb{R}_{\geq 0}^{(E(H))}$ such that every $v \in V(H)$ satisfies $\text{weight}_v(\psi) \geq 1$.
- The fractional edge cover number is

$$\text{fc}(H) = \min \{ \text{weight}(\psi) \mid \psi \text{ is a fractional edge cover of } H \} \in \mathbb{R}_{\geq 0} \cup \{\infty\}.$$

Observe that a finite hypergraph H satisfies $\text{fc}(H) = \infty$ if, and only if, H is not tame.

Remark 5.3.3 For a finite tame hypergraph H the minimum $\text{fc}(H)$ exists and is rational.

Proof. $\text{fc}(H)$ is the solution of the following linear programming problem in the variables $\psi(h)$, $h \in E(H)$: Minimise

$$\text{weight}(\psi) = \sum_{h \in E(H)} \psi(h)$$

subject to the conditions

$$\begin{aligned} \psi(h) &\geq 0 && \text{for all } h \in E(H) \\ \text{weight}_v(\psi) = \sum_{\substack{h \ni v \\ h \in E(H)}} \psi(h) &\geq 1 && \text{for all } v \in V(H) \end{aligned}$$

This problem has a solution because H is tame. It follows from standard results on linear programming that the solution is attained and that it is a rational number (since the coefficients are rationals). $\square$

Lemma 5.3.4 Let $H = (V, E)$ be a finite tame hypergraph.

1. An induced subhypergraph H' of H satisfies $\text{fc}(H') \leq \text{fc}(H)$.
2. $0 \leq \text{fc}(H) \leq |E|$.
3. $\frac{|V|}{\max_{h \in E} |h|} \leq \text{fc}(H)$.

Proof. 1,2: Easy. 3: Let ψ be a fractional edge cover of H . Then

$$\begin{aligned} |V| &= \sum_{v \in V} 1 \leq \sum_{v \in V} \sum_{h \ni v} \psi(h) = \sum_{h \in E} \sum_{v \in h} \psi(h) = \sum_{h \in E} \left(\psi(h) \cdot \sum_{v \in h} 1 \right) = \sum_{h \in E} \psi(h) \cdot |h| \\ &\leq \sum_{h \in E} \psi(h) \cdot \max_{h' \in E} |h'| = \left(\max_{h \in E} |h| \right) \cdot \sum_{h \in E} \psi(h). \end{aligned}$$

By Remark 5.3.3 there is a fractional edge cover ψ_0 with $\text{fc}(H) = \sum_{h \in E} \psi_0(h)$, and thus we have

$$\frac{|V|}{\max_{h \in E} |h|} \leq \sum_{h \in E} \psi_0(h) = \text{fc}(H).$$

□

Definition 5.3.5 We consider the following width function on hypergraphs H :

$$\begin{aligned} \text{fc}_H : \mathcal{P}_{<\omega}(V(H)) &\rightarrow \mathbb{R} \cup \{\infty\} \\ X &\mapsto \text{fc}(H[X]). \end{aligned}$$

We call $\text{fc}_H(X)$ the fractional cover number of X .

Obviously, for a finite hypergraph H we have $\text{fc}(H) = \text{fc}_H(V(H))$.

Lemma 5.3.6 Let H be a hypergraph and $X \subseteq V(H)$ a finite subset of $V(H)$. Then

$$\text{fc}_H(X) = \inf \{ \text{weight}(\varphi) \mid \varphi \in \mathbb{R}_{\geq 0}^{(E(H))}, \text{weight}_x(\varphi) \geq 1 \text{ for all } x \in X \}.$$

Proof. For $\varphi \in \mathbb{R}_{\geq 0}^{(E(H))}$ satisfying $\text{weight}_x(\varphi) \geq 1$ for all $x \in X$, define the function $\psi_\varphi \in \mathbb{R}_{\geq 0}^{(E(H[X]))}$ by

$$\psi_\varphi(e_X) = \sum_{\substack{e \in E(H) \\ e_X = e \cap X}} \varphi(e).$$

Note that

- $\text{weight}_x(\psi_\varphi) = \text{weight}_x(\varphi) \geq 1$ for every $x \in X$, and thus ψ_φ is a fractional edge cover of $H[X]$, and
- $\text{weight}(\psi_\varphi) = \text{weight}(\varphi)$.

Conversely, choose a function $\tau : E(H[X]) \rightarrow E(H)$ satisfying $e_X = \tau(e_X) \cap X$. For a fractional edge cover $\psi \in \mathbb{R}_{\geq 0}^{(E(H[X]))}$ define $\varphi_\psi \in \mathbb{R}_{\geq 0}^{(E(H))}$ by

$$\varphi_\psi(e) = \begin{cases} \psi(e_X) & \text{if } e = \tau(e_X) \text{ for an edge } e_X \in E(H[X]), \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Note that

- $\text{weight}_x(\varphi_\psi) = \text{weight}_x(\psi) \geq 1$ for every $x \in X$, and
- $\text{weight}(\varphi_\psi) = \text{weight}(\psi)$.

Then we have

$$\begin{aligned}
\text{fc}_H(X) &= \inf \{ \text{weight}(\psi) \mid \psi \in \mathbb{R}_{\geq 0}^{E(H[X])}, \text{weight}_x(\psi) \geq 1 \text{ for all } x \in X \} \\
&= \inf \{ \text{weight}(\varphi_\psi) \mid \psi \in \mathbb{R}_{\geq 0}^{E(H[X])}, \text{weight}_x(\psi) \geq 1 \text{ for all } x \in X \} \\
&\geq \inf \{ \text{weight}(\varphi) \mid \varphi \in \mathbb{R}_{\geq 0}^{E(H)}, \text{weight}_x(\varphi) \geq 1 \text{ for all } x \in X \} \\
&\geq \inf \{ \text{weight}(\psi_\varphi) \mid \varphi \in \mathbb{R}_{\geq 0}^{E(H)}, \text{weight}_x(\varphi) \geq 1 \text{ for all } x \in X \} \\
&\geq \inf \{ \text{weight}(\psi) \mid \psi \in \mathbb{R}_{\geq 0}^{E(H[X])}, \text{weight}_x(\psi) \geq 1 \text{ for all } x \in X \} \\
&= \text{fc}_H(X).
\end{aligned}$$

□

The following result is based on an idea of M. Weyer. For every real number $r > 0$ we construct a hypergraph H_r such that $\text{fc}_H\text{-ghw}(H) = r$. Of course, if r is irrational then H_r must be infinite by Remark 5.3.3.

Example 5.3.7 Let $1 \leq k \leq n$ be integers. Consider the hypergraph $H_{\frac{n}{k}}$ given by

$$H_{\frac{n}{k}} = (\{1, \dots, n\}, \mathcal{P}_{=k}(\{1, \dots, n\})),$$

consisting of n vertices and all k -element subsets as hyperedges. Then

$$\text{fc}_{H_{\frac{n}{k}}}\text{-ghw}(H_{\frac{n}{k}}) = \frac{n}{k}.$$

Proof. Note that since $\underline{H}$ is a complete graph, every tree decomposition (T, B) of $\underline{H}$ contains a piece $B_t = V(H)$, and thus $\text{fc}_{H_{\frac{n}{k}}}\text{-ghw}(H_{\frac{n}{k}}) = \text{fc}_{H_{\frac{n}{k}}}(V(H_{\frac{n}{k}})) = \text{fc}(H_{\frac{n}{k}})$. We first show that $\text{fc}(H_{\frac{n}{k}}) \leq \frac{n}{k}$, by defining a fractional edge cover with weight $\frac{n}{k}$. Note that $|E(H_{\frac{n}{k}})| = \binom{n}{k}$, and that each $v \in V(H_{\frac{n}{k}})$ is contained in exactly $\binom{n-1}{k-1}$ hyperedges of $E(H_{\frac{n}{k}})$. Define $\psi : E(H) \rightarrow [0, \infty)$ by

$$\psi(e) = \frac{1}{\binom{n-1}{k-1}},$$

for all $e \in E(H)$. Then each $v \in V(H_{\frac{n}{k}})$ satisfies $\sum_{e \ni v} \psi(e) = 1$, and hence ψ is a fractional edge cover for $H_{\frac{n}{k}}$. The weight of ψ is

$$\text{weight}(\psi) = \sum_{e \in E(H_{\frac{n}{k}})} \psi(e) = \binom{n}{k} \cdot \frac{1}{\binom{n-1}{k-1}} = \frac{n!}{k!(n-k)!} \cdot \frac{(k-1)!(n-k)!}{(n-1)!} = \frac{n}{k}.$$

Therefore $\text{fc}(H_{\frac{n}{k}}) \leq \frac{n}{k}$. Conversely, by Lemma 5.3.4 we have

$$\text{fc}(H_{\frac{n}{k}}) \geq \frac{|V(H)|}{\max \{ |e| \mid e \in E(H_{\frac{n}{k}}) \}} = \frac{n}{k}.$$

□

Corollary 5.3.8 For every $r \in \mathbb{R}$ there is a hypergraph H_r with

$$\text{fc}_{H_r}\text{-ghw}(H_r) = r.$$

Proof. Choose a sequence $(q_n)_{n < \omega}$ of rational numbers satisfying $q_n < r$ such that $\lim_{n \rightarrow \infty} q_n = r$. Define H_r to be the disjoint union

$$H_r = \bigcup_{n < \omega} H_{q_n},$$

where each H_{q_n} is as in Example 5.3.7. It is easy to see that $\text{fc}_{H_r}\text{-ghw}(H_r) = r$. □

5.3.2 Properties of fc_H

Lemma 5.3.9 *For a hypergraph H , the width function fc_H of H is monotone, additive and weakly submodular. If H is tame, then so is fc_H .*

Proof. Monotonicity: Suppose X, Y are finite sets such that $X \subseteq Y \subseteq V(H)$. By Lemma 5.3.6 we have

$$\begin{aligned} \text{fc}_H(X) &= \inf \{ \text{weight}(\varphi) \mid \varphi \in \mathbb{R}_{\geq 0}^{(E(H))}, \text{weight}_x(\varphi) \geq 1 \text{ for all } x \in X \} \\ &\leq \inf \{ \text{weight}(\varphi) \mid \varphi \in \mathbb{R}_{\geq 0}^{(E(H))}, \text{weight}_y(\varphi) \geq 1 \text{ for all } y \in Y \} \\ &= \text{fc}_H(Y). \end{aligned}$$

Weak submodularity: Let H be a hypergraph and let X, Y be finite subsets of $V(H)$. By Lemma 5.3.6 we have

$$\begin{aligned} \text{fc}_H(X \cup Y) &= \inf \{ \text{weight}(\varphi) \mid \varphi \in \mathbb{R}_{\geq 0}^{(E(H))}, \text{weight}_z(\varphi) \geq 1 \text{ for all } z \in X \cup Y \} \\ &\leq \inf \{ \text{weight}(\varphi_X + \varphi_Y) \mid \varphi_X, \varphi_Y \in \mathbb{R}_{\geq 0}^{(E(H))}, \\ &\quad \text{weight}_x(\varphi_X) \geq 1 \text{ and } \text{weight}_y(\varphi_Y) \geq 1 \text{ for all } x \in X, y \in Y \} \\ &= \inf \{ \text{weight}(\varphi_X) \mid \varphi \in \mathbb{R}_{\geq 0}^{(E(H))}, \text{weight}_x(\varphi_X) \geq 1 \text{ for all } x \in X \} \\ &\quad + \inf \{ \text{weight}(\varphi_Y) \mid \varphi \in \mathbb{R}_{\geq 0}^{(E(H))}, \text{weight}_y(\varphi_Y) \geq 1 \text{ for all } y \in Y \} \\ &= \text{fc}_H(X) + \text{fc}_H(Y). \end{aligned}$$

Additivity: By weak submodularity we only have to show that

$$\text{fc}_H(X \cup Y) \geq \text{fc}_H(X) + \text{fc}_H(Y)$$

for finite, non-touching sets $X, Y \subseteq V(H)$. For $\varphi \in \mathbb{R}_{\geq 0}^{(E(H))}$ with $\text{weight}_z(\varphi) \geq 1$ for all $z \in X \cup Y$, define $\varphi_X, \varphi_Y \in \mathbb{R}_{\geq 0}^{(E(H))}$ by

$$\varphi_X(e) = \begin{cases} \varphi(e) & \text{if } e \cap X \neq \emptyset, \text{ and} \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\varphi_Y(e) = \begin{cases} \varphi(e) & \text{if } e \cap Y \neq \emptyset, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Then for $x \in X$ we have

$$\text{weight}_x(\varphi_X) = \sum_{x \in e \in E(H)} \varphi_X(e) = \sum_{x \in e \in E(H)} \varphi(e) \geq 1,$$

and similarly for $y \in Y$ and φ_Y . Since by assumption,

$$\{e \in E(H) \mid e \cap (X \cup Y) \neq \emptyset\} = \{e \in E(H) \mid e \cap X \neq \emptyset\} \dot{\cup} \{e \in E(H) \mid e \cap Y \neq \emptyset\},$$

we have $\varphi \geq \varphi_X + \varphi_Y$. Thus,

$$\begin{aligned} \text{fc}_H(X \cup Y) &= \inf \{ \text{weight}(\varphi) \mid \varphi \in \mathbb{R}_{\geq 0}^{(E(H))}, \text{weight}_z(\varphi) \geq 1 \text{ for all } z \in X \cup Y \} \\ &\geq \inf \{ \text{weight}(\varphi_X + \varphi_Y) \mid \varphi_X, \varphi_Y \in \mathbb{R}_{\geq 0}^{(E(H))}, \\ &\quad \text{weight}_x(\varphi_X) \geq 1 \text{ and } \text{weight}_y(\varphi_Y) \geq 1 \text{ for all } x \in X, y \in Y \} \\ &\geq \inf \{ \text{weight}(\psi) \mid \psi \in \mathbb{R}_{\geq 0}^{(E(H))}, \text{weight}_x(\psi) \geq 1 \text{ for all } x \in X \} \\ &\quad + \inf \{ \text{weight}(\psi') \mid \psi' \in \mathbb{R}_{\geq 0}^{(E(H))}, \text{weight}_y(\psi') \geq 1 \text{ for all } y \in Y \} \\ &= \text{fc}_H(X) + \text{fc}_H(Y). \end{aligned}$$

Tameness: If H has no isolated vertices, then every vertex v is contained in a hyperedge h . Now clearly $\text{fc}_H(h) = 1$. $\square$

Corollary 5.3.10 *For a finite hypergraph H , the width function fc_H on H satisfies Theorem 2.4.23,1 and 2. If H is tame, then fc_H also satisfies Theorem 2.4.23,3.*

Proof. Use that $\text{fc}_H = \text{fc}_H^{\text{mon}}$. $\square$

Remark 5.3.11 *Let H be a hypergraph. Let $X \subseteq V(H)$ be a finite subset of $V(H)$. Then $\text{fc}_H(X) \leq c_H(X)$.*

Proof. Suppose $c_H(X) \leq k$. Then there is a subset $E_X \subseteq E(H[X])$ satisfying $|E_X| \leq k$ and $X = \bigcup E_X$. Therefore the function $\psi \in \mathbb{R}_{\geq 0}^{E(H[X])}$ given by

$$\psi(e) = \begin{cases} 1 & \text{if } e \in E_X, \text{ and} \\ 0 & \text{otherwise} \end{cases}$$

is a fractional edge cover of $H[X]$ satisfying $\text{weight}(\psi) = \sum_{e \in E(H[X])} \psi(e) = k$. $\square$

Remark 5.3.12 *Any hypergraph H satisfies*

$$\text{fc}_H - \text{hw}(H) = \text{fc}_H - \text{ghw}(H) \leq c_H - \text{ghw}(H) \leq c_H - \text{hw}(H).$$

Proof. This follows from Remark 5.3.11 and monotonicity of fc_H . $\square$

By Theorem 2.4.23,3, for a tame hypergraph H and $k \in \mathbb{R}$ we have

$$\text{fc}_H - \text{cw}(H) \leq k \implies \text{fc}_H - \text{hw}(H) \leq 3k + 2.$$

This reproves a result of M. Grohe and D. Marx in [GM05]. They also show that fc_H -hypertree-width and c_H -hypertree-width are not coherent:

Fact 5.3.13 (Grohe, Marx) *For every integer $n > 0$ there is a hypergraph H_n with $\text{fc}_{H_n} - \text{hw}(H_n) \leq 2$ and $c_{H_n} - \text{hw}(H_n) = n$.*

If we only consider classes of hypergraphs that are bounded by some fixed integer $\kappa > 0$, then we get linear coherence:

Proposition 5.3.14 *Let H be a hypergraph bounded by $\kappa > 0$.*

1. *If H is tame, then $\text{fc}_H - \text{hw}(H) \leq \text{card} - \text{hw}(H) \leq \kappa \cdot (\text{fc}_H - \text{hw}(H))$.*
2. *$\text{fc}_H - \text{hw}(H) \leq c_H - \text{hw}(H) \leq \kappa \cdot (\text{fc}_H - \text{hw}(H))$.*

Proof. 1: Since fc_H is monotone, tame, and weakly submodular, by Remark 2.4.5, 4 we have $\text{fc}_H - \text{hw}(H) \leq \text{card} - \text{hw}(H)$. For the second inequality, by Remark 5.3.12 it suffices to show that $\text{card} - \text{hw}(H) \leq \kappa \cdot (\text{fc}_H - \text{ghw}(H))$. Let (T, B, B) be a generalised hypertree decomposition of H with fc_H -width $(T, B, B) \leq k$. Then every node $t \in T$ satisfies $\text{fc}_H(B_t) \leq k$. Thus with Lemma 5.3.4, 3, we get

$$\frac{|B_t|}{\kappa} \leq \frac{|B_t|}{\max_{e \in E(H[B_t])} |e|} \leq \text{fc}(H[B_t]) = \text{fc}_H(B_t) \leq k.$$

Therefore, for every $t \in T$ we have $|B_t| \leq \kappa \cdot k$, and (T, B, B) witnesses that $\text{card} - \text{hw}(H) \leq \kappa \cdot k$.

2: By Remark 5.3.12 we have $\text{fc}_H - \text{hw}(H) \leq c_H - \text{hw}(H)$. If H is not tame, then $c_H - \text{hw}(H) = \text{fc}_H - \text{hw}(H) = \infty$, so we now assume that H is tame. Then the second inequality is proved by

$$c_H - \text{hw}(H) \leq \text{card} - \text{hw}(H) \leq \kappa \cdot (\text{fc}_H - \text{ghw}(H)),$$

which follows from 1 and 3.1.15, 1. $\square$

In [GM05, Theorem 5], M. Grohe and D. Marx proved the following fact.

Fact 5.3.15 *Let $k \in \mathbb{R}$ and let f be the family of width functions fc_H , for all hypergraphs H . Let $\mathcal{C}_k^f = \{\text{structures } \mathcal{M} \mid \text{fc}_{H_{\mathcal{M}}}(H_{\mathcal{M}}) \leq k\}$. Then $\text{HOM}_{\text{enum}}(\mathcal{C}_k^f, -) \in \text{P}$.*

It is an open problem, whether $\text{SUBSTR}_{\text{enum}}^{\mathcal{C}_k^f} \in \text{P}$.

Corollary 5.3.16 *Let f be the family of width functions fc_H , for all hypergraphs H . Then $\text{HOM}(\text{DECOMPOSED}^{\mathcal{C}_k^f}, -) \in \text{P}$.*

Proof. Use Theorem 2.3.18. $\square$

Corollary 5.3.17 (Compactness of fc_H -ghw) *Let H be a hypergraph such that fc_H has finite character. For example, this is the case if H is bounded. Then*

$$\begin{aligned} \text{fc}_H - \text{ghw}(H) &\leq k \\ \iff \\ \text{fc}_{H_0} - \text{ghw}(H_0) &\leq k \text{ for all finite induced subhypergraphs } H_0 \text{ of } H. \end{aligned}$$

Proof. Use Remark 2.1.17 and Theorem 4.2.1. If H is bounded, then fc_H has finite character by Lemma 5.3.4 and Example 4.1.2, 2. $\square$

5.3.3 Fractional edge covers and directed hypergraphs

In analogy to $\overrightarrow{\text{card}}_{\vec{G}}$ and $\vec{c}_{\vec{H}}$, we can define the width function $\overrightarrow{\text{fc}}_{\vec{H}}$ on a directed hypergraph $\vec{H}$. More generally, for a width function f on the underlying graph $\underline{\vec{H}}$ of $\vec{H}$ we can define a width function $\vec{f}_{\vec{H}}$ as follows.

Definition 5.3.18 *Let $\vec{H}$ be a directed hypergraph, and let f be a width function on the underlying graph $\underline{\vec{H}}$ of $\vec{H}$. Then the width function $\vec{f}_{\vec{H}}$ is given by*

$$\begin{aligned} \vec{f}_{\vec{H}} : \mathcal{P}_{<\omega}(V(\vec{H})) &\rightarrow \mathbb{R} \cup \{\infty\}, \\ X &\mapsto \inf\{f(Y) \mid Y \subseteq V(\vec{H}), X = \langle Y \rangle\}. \end{aligned}$$

Combining the examples of Example 5.2.12 and Fact 5.3.13, it is easy to find a class $\mathcal{C}$ of directed hypergraphs such that the $\vec{c}_{\vec{H}}$ -hypertree-width and the fc_H -hypertree-width of $\mathcal{C}$ are unbounded, but the $\overrightarrow{\text{fc}}_{\vec{H}}$ -hypertree-width is bounded.

Recall that card is monotone, but $\overrightarrow{\text{card}}_{\vec{G}}$ is not. Thus f being monotone does not imply that $\vec{f}_{\vec{H}}$ is monotone. Moreover, for a directed hypergraph $\vec{H}$ we have already seen, that—although the width function $c_{\underline{\vec{H}}}^{\text{mon}}$ is additive (Lemma 3.1.6)—the width function $\vec{c}_{\vec{H}}^{\text{mon}}$ need not be additive (cf. Example 5.1.11). Submodularity and tameness are passed on:

Lemma 5.3.19 *Let $\vec{H}$ be a directed hypergraph, and let f be a monotone width function on the underlying graph $\underline{\vec{H}}$ of $\vec{H}$.*

1. If f is weakly submodular, then so is $\vec{f}_{\vec{H}}^{\text{non}}$.
2. If f is tame, then so is $\vec{f}_{\vec{H}}^{\text{non}}$.

Proof. 1: Let f be weakly submodular, and let X_1, X_2 be finite subsets of $V(\vec{H})$. For $\varepsilon > 0$ choose $Y_1, Y_2 \subseteq V(\vec{H})$ such that $X_i \subseteq \langle Y_i \rangle$ and $\vec{f}_{\vec{H}}(X_i) + \varepsilon \geq f(Y_i)$ for $i = 1, 2$. Then

$$\vec{f}_{\vec{H}}(X_1) + \vec{f}_{\vec{H}}(X_2) + 2\varepsilon \geq f(Y_1) + f(Y_2) \geq f(Y_1 \cup Y_2) \geq \vec{f}_{\vec{H}}(X_1 \cup X_2),$$

where the second inequality holds because f is weakly submodular, and the last inequality holds because $X_1 \cup X_2 \subseteq \langle Y_1 \cup Y_2 \rangle$.

2: Obvious. □

Corollary 5.3.20 *For a finite directed hypergraph $\vec{H}$, the width function $\vec{fc}_{\vec{H}}$ on $\vec{H}$ satisfies Theorem 2.4.23, 1 and 2.* □

Example 5.1.11 also shows that $\vec{fc}_{\vec{H}}$ is not additive in general.

It is easy to see that the tractability results from Section 2.3 are passed on from C_k^f to $C_k^{\vec{f}}$. In particular, for the family $\vec{f}$ of width functions $\vec{f}_{\vec{H}}$ for every directed hypergraph $\vec{H}$, we have $\text{HOM}(\text{DECOMPOSED}^{C_k^{\vec{f}}}, -) \in \mathbf{P}$.

Epilogue

The focus of this thesis is on generalised hypertree-width, its generalisations and related¹ invariants. Of the related invariants known so far, cop-width and hypertree-width are closest to generalised hypertree-width, bounding it from below and from above. Therefore they were also examined in detail. Two very basic questions had to be left open even in the finite case:

- Under what conditions is $f\text{-bramble-no}(\underline{H}) = f\text{-cw}(\underline{H})$? Is $f = c_H^{\text{mon}}$ sufficient?
- $c_H\text{-ghw}(\underline{H})$ and $c_H\text{-hw}(\underline{H})$ are linearly coherent by [AGG05]. Is there a natural class of equipped graphs (G, f) (larger than that of those of the form $(\underline{H}, c_H^{\text{mon}})$) such that $f\text{-ghw}(G)$ and $f\text{-hw}(G)$ are (linearly) coherent?

The infinite case is much more complicated. Open questions include:

- Is it true that in the games $\text{RC}_{\text{mon}}(G, f, k)$ and $\text{RC}(G, f, k)$, one of the players must have a winning strategy?
- Is it true that if the cops have a winning strategy for $\text{RC}(G, f, k)$, then they also have a positional winning strategy?
- Is it true that if the robber has a winning strategy for $\text{RC}_{\text{mon}}(G, f, k)$ or $\text{RC}(G, f, k)$, then he also has a positional winning strategy? (This should be easy for $\text{RC}_{\text{mon}}(G, f, k)$.)
- How about compactness properties for the other invariants defined in Section 2.4?
- Is $f\text{-ghw}(G) = k$ always witnessed by a single generalised hypertree decomposition? (This is true if f only takes values in a well-ordered set such as $\omega \cup \{\infty\}$. But how about fractional hypertree-width?)
- If H is locally bounded, does it follow that fc_H has finite character?
- **Conjecture.** If $\underline{H}$ contains no infinite complete subgraph, then c_H has finite character.

Tree decompositions of infinite graphs

There is more than one way of defining tree decompositions for infinite graphs. For this thesis we have settled on one of them.

P. D. Seymour and R. Thomas in [ST93] have a more restrictive definition, the only difference being that the tree T in a tree decomposition (T, B) must be *rayless*, i. e. it must not contain a *ray* (a graph isomorphic to Figure 2.2, B) as a subgraph. Apart from the fact that a ray is a tree which does not have a tree decomposition

¹In the sense of being linearly coherent.

in this sense (and apart from the necessary failure of the compactness property), this approach seems to work remarkably well. If we had followed this line, then of course we would have had to adapt some other definitions as well. The winning condition in $\text{RC}(G, f, k)$ and $\text{RC}_{\text{mon}}(G, f, k)$ would have been much simpler: The robber wins every infinite play. Brambles and tangles would have been defined without condition (B3) and k -decomposability without condition (D3). It is not clear how we could have fixed the definition of k -trees.

We could also generalise our definition of tree decompositions, admitting infinite pieces B_t . But then we would have to consider width functions defined on arbitrary (possibly infinite) sets of vertices. In this context it would also be natural to permit infinite hyperedges in hypergraphs. Every attempt to define $\overrightarrow{\text{card}}_{\vec{G}}$ or $\vec{c}_H$ on infinite mixed graphs or infinite directed hypergraphs naturally leads to this generalisation.

There is also one point that only affects hypertree decompositions. In order for condition (HD3) to make sense we need T to be a directed tree. However, our definition of infinite directed trees is unnecessarily restrictive in that it requires a root. The root has no in-edge, while every other node has precisely one in-edge. But an infinite tree which is not rayless can also be directed in such a way that every node has precisely one in-edge. (Intuitively, the root of such a tree is the limit of an inverse ray.) Therefore, let us call an *ordered tree* every tree the edges of which are directed in such a way that every node has at most one in-edge.

Condition (HD3) makes sense if T is an ordered tree. Therefore we could say that (T, B, C) is a *weak hypertree decomposition* if T is an ordered tree (not necessarily a directed tree), and T satisfies (TD1), (TD2), (TD3), (HD2), (HD3). On finite graphs every weak hypertree decomposition is of course a hypertree decomposition. But the hypergraph H from Theorem 4.4.2, which has $c_H\text{-hw}(H) = 4$, has a weak hypertree decomposition of c_H -width 3. Thus it seems likely that the resulting notion of ‘weak hypertree-width’ is compact for locally bounded graphs. On the other hand, it seems to be hard to find a game characterisation for this new notion, unless along the lines of: ‘First, the robber player chooses a finite induced subhypergraph’.

The homomorphism problem

Of course one would like to have an analogue of M. Grohe’s classification theorem Fact 2.3.15 in terms of the underlying hypergraph rather than the underlying graph. This seems to be rather hard, since the proof of the theorem uses the deep Excluded Grid Theorem from [RS86b]. We are still far from having an analogous theorem for hypergraphs. In fact, it is not even clear what a good definition of hypergraph minors should be.

Going in a different direction, one could try to find an analogue of Fact 2.3.15 in terms of the underlying mixed graph of a structure with relations and unary functions, with $\overrightarrow{\text{card}}_{\vec{G}\text{-ghw}}$ instead of tw .

References

- [AGG05] I. Adler, G. Gottlob, M. Grohe, Hypertree-Width and related Hypergraph Invariants. EuroComb 2005, accepted for publication.
- [Ad02] I. Adler. Diplomarbeit: Spiele als Hilfsmittel zu Strukturuntersuchungen bei Graphen und Hypergraphen. Freiburg 2002.
http://www.math.uni-freiburg.de/archiv/diplom/isolde_adler.html
- [Ad04] I. Adler. Marshals, Monotone Marshals, and Hypertree-Width. *J. of Graph Theory*, 47(4):275–296, 2004.
(This paper contains the main results of [Ad02].)
- [Ad05a] I. Adler. Tree-related Widths of Graphs and Hypergraphs, 2005. Submitted for publication.
- [Ad05b] I. Adler. Directed Tree-Width Examples. Submitted for publication.
- [AP81] S. Arnborg, A. Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k -trees. *Discr. Appl. Math.*, 23:11–24, 1981.
- [AP86] S. Arnborg, A. Proskurowski. Characterization and recognition of partial 3-trees. *SIAM J. Alg. Disc. Meth.* 7(2):305–314, 1986.
- [BDHK05] D. Berwanger, A. Dawar, P. Hunter, S. Kreutzer. DAG-width and parity games. 2005. Accepted for publication
- [BP71] L. W. Beineke, R. E. Pippert. Properties and characterizations of k -trees, *Mathematika*, 18:141–151. 1971.
- [CD05] H. Chen, V. Dalmau. Beyond hypertree width: Decomposition methods without decompositions. In: *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming*. 2005.
- [CJG05] D. Cohen, P. Jeavons, M. Gyssens, A Unified Theory of Structural Tractability for Constraint Satisfaction and Spread Cut Decomposition. 2005. *Proceedings of IJCAI’05* pp:72–77.
- [DKV02] V. Dalmau, Ph. G. Kolaitis, M. Y. Vardi. Constraint satisfaction, bounded tree-width and finite variable logics. In P. Van Hentenryck, editor, *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*. LNCS 2470:310–326. Springer 2002.
- [DF99] R. G. Downey, M. R. Fellows. *Parametrized Complexity*. Springer 1999.

- [Di97] R. Diestel. Graph Theory, Springer 1997.
- [EF99] H.-D. Ebbinghaus, J. Flum. Finite Model Theory. Springer 1999.
- [FFG02] J. Flum, M. Frick, M. Grohe. Query evaluation via tree-decompositions. J. of the ACM 49(6):716–752, 2002.
- [GGMSS05] G. Gottlob, M. Grohe, N. Musliu, M. Samer, F. Scarcello. Hypertree Decompositions: Structure, Algorithms, and Applications. Proc. of the 31st Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG'05), LNCS ???, Springer, ???.
- [GJ79] M. R. Garey, D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman 1979.
- [GLS01a] G. Gottlob, N. Leone, F. Scarcello. Hypertree Decompositions: A survey. In J. Sgall, A. Pultr, and P. Kolman (eds.): MFCS 2001, LNCS 2136:37–57. Springer 2001. .
- [GLS01b] G. Gottlob, N. Leone, F. Scarcello. Robbers, Marshals, and Guards: Game Theoretic and Logical Characterizations of Hypertree Width. Proc. of the 20th ACM Symposium on Principles of Database Systems:21–32. ACM Press 2001.
- [GLS01c] G. Gottlob, N. Leone, F. Scarcello. A Comparison of Structural CSP Decomposition Methods, Artificial Intelligence (2000),124(2):243–282. Preliminary version in IJCAU'99.
- [GLS02] G. Gottlob, N. Leone, F. Scarcello. Hypertree Decompositions and Tractable Queries. J. Computer and System Sciences 64(3):579–627, 2002. Preliminary version in PODS'99.
- [GM05] M. Grohe, D. Marx. Constraint Solving via fractional edge covers, to appear in Proc. of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006).
- [Gr03] M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. In Proc. of the 44th IEEE Symposium on Foundations of Computer Science (FOCS '03):552–561, 2003.
- [Grä] G. Grätzer. Universal Algebra. Van Nostrand, 1968,
- [Ha64] R. Halin. Über simpliziale Zerfällungen beliebiger (endlicher und unendlicher) Graphen. Mathematische Annalen 156:216–225, 1964.
- [Ha76] R. Halin. S-functions for graphs. J. Geometry 8:171–186, 1976.
- [JRST01] T. Johnson, N. Robertson, P. D. Seymour, R. Thomas. Directed Tree-Width. J. Comb. Theory (Series B) 82:138–154, 2001.
- [KT91] I. Kříž, R. Thomas. The Menger-like Property of the tree-width of Infinite Graphs. J. of Combinatorial Theory (Series B) 52:86–91, 1991.
- [KV95] Ph. G. Kolaitis, M. Y. Vardi. On the expressive power of datalog: tools and a case study. Journal of Computer and System Sciences 51(1):110–134, 1995.
- [LS99] A. Lustig, O. Shmueli. Acyclic Hypergraph projections. J. of Algorithms 30:400–422, 1999.

- [Re97] B. A. Reed. Tree-Width and Tangles: A New Connectivity Measure And Some Applications. In *Surveys in Combinatorics* (R. Bailey, ed.). Cambridge University Press 1997
- [Re99] B. Reed. Introducing Directed Tree-Width. 6th Twente Workshop on Graphs and Combinatorial Optimization (Enschede 1999). *Electronic Notes Discrete Math.* 3:8 pages (electronic). Elsevier 1999.
- [Ro70] D. Rose. Triangulated graphs and the elimination process, *J. Math. Analysis Appl.*, 32:597–609, 1970.
- [Ro74] D. Rose. On simple characterizations of k-trees, *Discrete Math.*, vol. 7:317–322, 1974 .
- [RS86a] N. Robertson, P. D. Seymour. Graph Minors. II. Algorithmic Aspects of Tree-Width. *J. of Algorithms*, 7:309–322, 1986.
- [RS86b] N. Robertson, P. D. Seymour. Graph Minors. V. Excluding a planar graph. *J. of Combinatorial Theory (Series B)*, 41:92–114, 1986.
- [RS91] N. Robertson, P. D. Seymour. Graph Minors. X. Obstructions to Tree-Decompositions. *J. of Combinatorial Theory (Series B)* 52:153–190, 1991.
- [Sa85] D. Saccà. Closures of Database Hypergraphs. *J. of the Association for Computing Machinery*, 32(4):774–803, 1985.
- [ST93] P. D. Seymour and R. Thomas. Graph Searching and a Min-Max Theorem for Tree-Width. *J. of Combinatorial Theory (Series B)* 58:22–33, 1993.
- [Th] R. Thomas. The tree-width compactness theorem for hypergraphs. Unpublished manuscript.
- [Thsen89] C. Thomassen. Configurations in graphs of large minimum degree, connectivity or chromatic number. In *Combinatorial Mathematics, Proc. of the 3rd International Conf., New York 1985*, *Annals of the New York Academy of Science*, 555:402–412, 1989.
- [Ya81] M. Yannakakis. Algorithms for acyclic database schemes. In *7th International Conf. on Very Large Data Bases*, 82–94, 1981.

Index

Symbols

B_{T_i}	6
$G \cup \hat{H}^v$	10
$G \times n$	60
$G_0 \subseteq G$	4
$G_1 \cap G_2$	4
$G_1 \cup G_2$	4
$H \cdot n$	60
$H \upharpoonright X$	34
$H^{\leq k}$	74
R_π	24
T_t	6
CQ	39
CS^{C_1}	38
$\text{RC}(G, f, k)$	22
$\text{RC}_{\text{mon}}(G, f, k)$	22
ΣH	59
β_t	28
card	16
c_H	17, 102
$\langle X \rangle$	87
$\text{COREDECOMPOSABLE}_{\text{ghd}}^{C_1}$	35
$\text{DECOMPOSABLE}_{\text{hd}}^{C_1}$	36
DECOMPOSED^{C_1}	37
$\text{fc}(H)$	101
fc_H	102
FPT	35
$\text{card}_{\vec{G}}$	88
GHW^k	74
$\hat{G}^v$	10
HD^{C_1}	36
$H_{\mathcal{M}}$	32
$\text{HOM}(\mathcal{C}, \mathcal{D})$	32
$\text{HOM}_{\text{enum}}(\mathcal{C}_1, _)$	34
HSP	73
$\lambda_{t,s}$	47
$\mathcal{M} \xrightarrow{\text{hom}} \mathcal{N}$	32
$\mathbb{R}_{\geq 0}^{(E(H))}$	101
$\mathcal{C}_k^f$	34, 57, 100
$\text{pred}(s)$	4
$\text{SUBSTR}_{\text{enum}}^{C_1}$	34
$\text{tw}(G)$	4
$\underline{H}$	17
$\text{var}(\varphi)$	40

$\varphi(\mathcal{N}) \neq \emptyset$	39
$\vec{G}$	87
$\text{width}(T, B)$	4
$\text{weight}(\psi)$	101
$\text{weight}_v(\psi)$	101
$\text{W}[1]$	35
f -bramble-no	44
f -cw(G)	25
f -cw _{mon} (G)	26
f -ghw(G)	16
f -hw(G)	16
f -link(G)	50
f -order($\mathcal{B}$)	44
f -branch-width(G)	46
f -ehw(G)	14
f -ghw(H)	18
f -hw(H)	18
f -tangle-no	45
f -width(T, B)	14
f -width(T, B, C)	15
f -width(T, κ, λ)	46
f_{mon}	20
$f_{G'}$	19
f_{C_1}	34
k -decomposable	84

A

acyclic	32, 72
additive width function	42
arcs	87

B

(B1), (B2), (B3)	44
(B2')	45
balanced f -separator	49
(BD1), (BD2), (BD3), (BD4)	46
M -big	49
bounded hypergraph	59
bramble	44
f -bramble-number	44
branch	4
branch decomposition	46
f -branch-width	46
f -branch-width'	47

- C**
- (C1) 22
 - (C2) 24
 - chord 6
 - chordal 6
 - clique 4
 - k -clique 4
 - coherent 41
 - complete graph 4
 - conjunctive query 39
 - connected 4, 17
 - connected component 4
 - construction tree 12
 - cop player 23
 - cop strategy 24
 - f -cop-width 25
 - core 33
 - covers 4
 - cycles 4
- D**
- (D1), (D2), (D3) 84
 - DAG-width 86
 - database 39
 - directed graph 87
 - directed hypergraph 94
 - directed tree 4
 - directed tree-width 86
- E**
- edges 3, 17, 87
 - equipped graph 15
 - escape space 24
 - exact f -hypertree-width 14
- F**
- finite character 76
 - forest 4
 - forward closure 87, 95
 - fractional cover number 102
 - fractional edge cover 101
 - fractional edge cover number 101
 - fractional hypertree-width 100
- G**
- Gaifman graph 17
 - generalised $\mathcal{C}_1$ -hypertree decomposition
34
 - generalised f -hypertree-width 16
 - generalised hypertree decomposition 15
 - graph 3
 - guard 15
- H**
- (HD1), (HD2), (HD3) 15
- homomorphism problem 40
 - hyperarcs 94
 - hyperedges 17, 94
 - hypergraph 17
 - hypergraph pair 54
 - Hypergraph Sandwich Problem ... 73
 - 'hypertree' 15
 - hypertree decomposition 15
 - $\mathcal{C}_1$ -hypertree decomposition 34
 - f -hypertree-width 16
- I**
- induced subgraph 4
 - induced subhypergraph 17
 - induced subhypergraph pair 54
- K**
- (KT1), (KT2), (KT3) 10
- L**
- linearly coherent 41
 - (f, k) -linked 49
 - f -linkedness 50
 - locally bounded hypergraph 79
- M**
- minors 5
 - mixed graph 87
 - monotone f -cop-width 26
 - monotone robber and cops game .. 22
 - monotone width function 19
- N**
- no promise algorithm 36
 - nodes 4
- O**
- f -order 44
 - ordered tree 109
- P**
- partial function symbols 31
 - partial subgraph 4
 - pebble game 33, 35
 - pieces 4
 - play 22
 - positional 25
 - predecessor 4
 - primal graph 17
 - prime graph 7
 - projection 73
 - promise algorithm 37
- R**
- (R1), (R2) 22

ray 85, 108
 rayless 108
 robber and cops game 22
 robber player 23
 robber strategy 25

S

(SD1), (SD2), (SD3) 7
 separating clique 7
 signature 31
 simplicial 20
 simplicial decomposition 7
 small 9
 smooth 9
 source set 94
 strictly monotone 24
 strongly coherent 41
 structural restrictions 32
 subcubic tree 46
 subgraph 4
 subhypergraph 17
 successor 4

T

tame hypergraph 17
 tame width function 42
 tangle 45
 f -tangle-number 45
 target set 94
 (TD1), (TD2), (TD3) 4
 thick 9
 touch 42
 touching triple 45
 tree 4
 tree component 28
 tree decomposition 4, 18
 k -tree 10
 tree-width 4, 18
 triangulation 6
 trivial hypergraph pair 62

U

underlying directed hypergraph ... 99
 underlying graph 17, 87
 underlying hypergraph ... 31, 32, 94
 underlying hypergraph of φ 40
 underlying mixed graph 99

V

vertices 3, 94

W

weak hypertree decomposition ... 109
 weakly submodular 42
 weight 101

width function 15, 88, 96
 f -width 15
 winning condition for $\text{RC}(G, f, k)$. 23
 winning strategy 25