Approximating fractional hypertree width

Dániel Marx[*]

Abstract

Fractional hypertree width is a hypergraph measure similar to tree width and hypertree width. Its algorithmic importance comes from the fact that, as shown in previous work [14], constraint satisfaction problems (CSP) and various problems in database theory are polynomial-time solvable if the input contains a bounded-width fractional hypertree decomposition of the hypergraph of the constraints. In this paper, we show that for every $w \geq 1$, there is a polynomial-time algorithm that, given a hypergraph H with fractional hypertree width at most w, computes a fractional hypertree decomposition of width $O(w^3)$ for H. This means that polynomial-time algorithms relying on bounded-width fractional hypertree decompositions no longer need to be given a decomposition explicitly in the input, since an appropriate decomposition can be computed in polynomial time. Therefore, if $\mathcal{H}$ is a class of hypergraphs with bounded fractional hypertree width, then CSP restricted to instances whose structure is in $\mathcal{H}$ is polynomial-time solvable. This makes bounded fractional hypertree width the most general known hypergraph property that makes CSP, Boolean Conjuctive Queries, and Conjunctive Query Containment polynomial-time solvable.

1 Introduction

Constraint satisfaction is a general framework that includes many standard algorithmic problems such as satisfiability, graph coloring, database queries, etc. A constraint satisfaction problem (CSP) consists of a set V of variables, a domain D, and a set C of constraints, where each constraint is a relation on a subset of the variables. The task is to assign a value from D to each variable such that every constraint is satisfied. For example, 3SAT can be interpreted as a CSP problem where the domain is $D = \{0, 1\}$ and the constraints in C correspond to the clauses (thus the arity of each constraint is 3). Certain fundamental problems in database theory, such as Boolean Conjunctive Queries and Conjunctive Query Containment, is equivalent to CSP. For more background, see e.g., [12, 5, 11, 16].

In general, solving constraint satisfaction problems is NP-hard if there are no additional restrictions on the instances. The main goal of the research on CSP is to identify tractable special cases of the general problem. The theoretical literature on CSP investigates two main types of restrictions. The first type is to restrict the *constraint language*, that is, the type of constraints that is allowed. This direction inlcudes the classical work of Schaefer [22] and its many generalizations. The second type is to restrict the *structure* induced by the constraints on the variables. The *hypergraph* of a CSP instance is defined to be a hypergraph on the variables of the instance such that for each constraint $c \in C$ there is a hyperedge e_c that contains all the variables that appear in c. If the hypergraph of the CSP instance has very simple structure, then the instance is easy to solve. For example, it is well-known that a CSP instance I with hypergraph H can be solved in time $\|I\|^{O(\mathrm{tw}(H))}$ [7], where $\mathrm{tw}(H)$ denotes the tree width of H and $\|I\|$ is the size of the representation of I in the input. Thus if we restrict the problem to instances where the tree width of the hypergraph is bounded by some constant w, then the problem is polynomial-time solvable. It is the goal of ongoing research to find other properties (besides bounded tree width) that make the problem polynomial-time tractable. Formally, for a class $\mathcal{H}$ of hypergraphs, let CSP($\mathcal{H}$) be the restriction of CSP where the hypergraph of the instance is assumed to be in $\mathcal{H}$. Our goal is to find and categorize classes $\mathcal{H}$ such that CSP($\mathcal{H}$) can be solved in polynomial time.

If the constraints have bounded arity (i.e., edge size in $\mathcal{H}$ is bounded by a constant), then the complexity of CSP($\mathcal{H}$) is well understood:

THEOREM 1.1. ([13, 15]) *Let* CSP($\mathcal{H}$) *contain all CSP instances whose underlying hypergraph is in $\mathcal{H}$. If $\mathcal{H}$ is a recursively enumerable class of hypergraphs with bounded edge size, then (assuming* FPT $\neq$ W[1]*)*

$$\text{CSP}(\mathcal{H}) \textit{ is polynomial-time solvable} \\ \Updownarrow \\ \mathcal{H} \textit{ has bounded tree width.}$$

The assumption FPT $\neq$ W[1] is a standard hypothesis of parameterized complexity. Thus in the bounded-arity case bounded tree width is the only property of the hypergraph that can make the problem polynomial-time solvable.

The situation is much less understood in the unbounded arity case, i.e., when there is no bound on the maximum edge size in $\mathcal{H}$. First, the complexity in the unbounded-arity case depends on how the constraints are represented.

[*]Budapest University of Technology and Economics, Budapest H-1521, Hungary (dmarx@cs.bme.hu). Research supported by the Magyary Zoltán Postdoc Fellowship and the Hungarian National Research Fund (Grant Number OTKA 67651).

In the bounded-arity case, if each constraint contains at most r variables (r being a fixed constant), then every reasonable representation of a constraint has size $|D|^{O(r)}$. Therefore, the size of different representations can differ only by a polynomial factor. On the other hand, if there is no bound on the arity, then there can be exponential difference between the size of succinct representations (e.g., formulas) and verbose representations (e.g., truth tables). The running time of an algorithm is expressed as a function of the input size, hence the complexity of the problem can depend on how the input is represented: longer representation means that it is potentially easier to obtain a polynomial-time algorithm.

The most well-studied representation of constraints is listing all the tuples that satisfy the constraint. In this case, the size of the representation of a constraint relation is proportional to the number of satisfying tuples. This representation is very natural in problems involving relational databases, where the constraints are database relations that are actually stored as a sequence of tuples. If we want to use results on CSP in a database-theoretic setting, then we have to consider this representation.

Unlike in the bounded-arity case, if there is no bound on the number of variables in a constraint, then bounded tree width is not the right structural criterion for the tractability of the problem. It remains true that an instance with hypergraph H can be solved in time $\|I\|^{O(\mathrm{tw}(H))}$. However, there are classes $\mathcal{H}$ of hypergraphs with unbounded tree width such that CSP($\mathcal{H}$) is polynomial-time solvable. A very simple example is the class that contains those hypergraphs where one of the edges cover all the vertices. If the hypergraph H of a CSP instance belongs to this class, then it is easy to solve: there is a constraint that contains every variable, thus all we have to do is enumerating the satisfying tuples of this constraint and checking whether there is a tuple among them that satisfies every other constraint. This idea can be generalized: if we restrict the problem to hypergraphs that can be covered by k edges (for some fixed constant k), then CSP can be solved by enumerating all the possible combinations of satisfying tuples for k constraints that cover all the variables. This observation motivated the definition of (generalized) hypertree width [9, 1, 8], which is defined similarly to tree width, but instead of the requirement that each bag contains a bounded number of vertices, we require that each bag can be covered by a bounded number of edges (see Section 2 for the precise definition). As shown in [9], CSP($\mathcal{H}$) is polynomial-time solvable if $\mathcal{H}$ has bounded (generalized) hypertree width.

In [14], new tractable classes $\mathcal{H}$ with unbounded hypertree width were identified. It was shown, using Shearer's Lemma [4], that a CSP instance has only a polynomial number of solutions and they can be enumerated efficiently if the hypergraph of the instance has bounded fractional edge cover number. Thus CSP($\mathcal{H}$) is polynomial-time solvable if $\mathcal{H}$ has bounded fractional edge cover number. Fractional hypertree width is defined analogously to generalized hypertree width, but now we only require that each bag has bounded fractional edge cover number. As shown in [14], if $\mathcal{H}$ is a class of hypergraphs with bounded fractional hypertree width, then CSP($\mathcal{H}$) can be solved in polynomial time, if the input contains a tree decomposition of the hypergraph of the instance with bounded fractional hypertree width. However, it remained an open question whether it is possible to find such a tree decomposition in polynomial time and whether CSP($\mathcal{H}$) (without any extra input) is polynomial-time solvable for such $\mathcal{H}$.

Our results. The main result of the paper is an algorithm that computes approximately optimal fractional hypertree decompositions. More precisely, we show that for every $w \geq 1$, there is a polynomial-time algorithm that, given a hypergraph H with fractional hypertree width at most w, computes a tree decomposition of H with fractional hypertree width $O(w^3)$ (Theorem 4.1). Therefore, if every hypergraph in $\mathcal{H}$ has fractional hypertree width at most w, then CSP($\mathcal{H}$) is polynomial-time solvable: For every instance, we can compute a tree decomposition with fractional hypertree width $O(w^3)$ and then use the algorithm of [14]. Thus our result makes bounded fractional hypertree width the strictly most general known hypergraph property that allows CSP to be solved in polynomial time. Figure 1 shows some of the known tractable hypergraph properties (note that the elements of this Venn diagram are sets of hypergraphs; e.g., the set "bounded tree width" contains every set $\mathcal{H}$ of hypergraphs with bounded tree width). All the inclusions in the figure are proper. The tractable classes for CSP translate to tractable classes for Boolean Conjunctive Queries and Conjunctive Query Containment [16], thus bounded fractional hypertree width is the most general known tractability criterion for those problems as well.

Algorithms for finding tree decompositions and characterization theorems for (generalizations of) tree width often follow a certain pattern. For example, the same high-level idea is used for tree width [6, Section 11.2], rank width [21, 19], hypertree width [1], and branch width of matroids and submodular functions [20]. Simplifying somewhat, this general pattern can be summarized the following way: We decompose the problem into two parts by finding a small balanced separation, a tree decomposition for each part is constructed using the algorithm recursively, and the tree decompositions for the parts are joined in an appropriate way to obtain a tree decomposition for the original problem. A balanced separation of a subset W is a partition (A, B) of W and a set S separating A and B, such that A and B are both small compared to W (the exact definition of small depends on the actual type of tree decomposition we are looking for). Depending on the approximation ratio and the running time we are trying to achieve, the problem of finding a

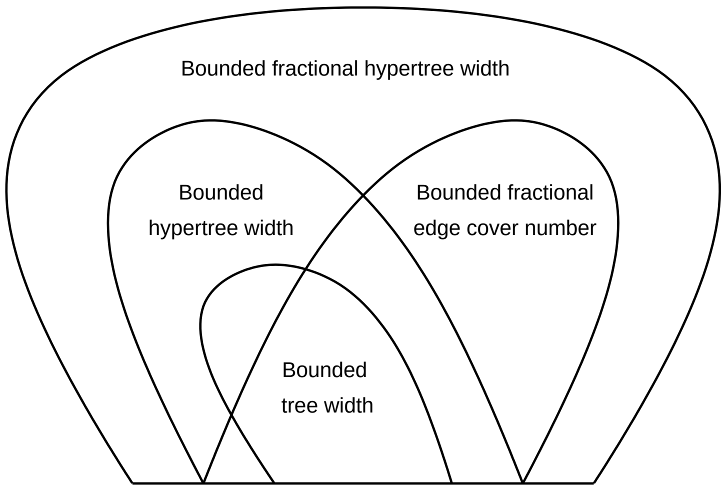
Bounded fractional hypertree width
Bounded hypertree width
Bounded fractional edge cover number
Bounded tree width

Figure 1: Hypergraph properties that make CSP polynomial-time solvable.

balanced separation is either reduced to a sparsest cut problem or (using brute force) it is reduced to the problem of finding a small (A, B)-separator, i.e., a set whose deletion disconnects A and B.

Can we use a similar approach for constructing fractional hypertree decompositions? With appropriate modifications, the recursive algorithm works for such decompositions as well (Section 4). The crucial question is how to find a balanced separation where S has small fraction edge cover number. Using brute force in a not completely trivial way, the search for a balanced separation can be reduced to finding an (A, B)-separator with small fractional edge cover number (Lemma 3.4). The main technical contribution of the paper is an approximation algorithm for finding such separators: if there is an (A, B)-separator with fractional edge cover number at most w, then the algorithm finds an (A, B)-separator with fractional edge cover number $O(w^3)$ (Section 3). The running time is polynomial for every fixed w.

For other types of tree decompositions, the corresponding (A, B)-separation problem can be solved using flow techniques, brute force, or submodularity. None of these techniques seem to be relevant when the goal is to minimize the fractional edge cover number of the separator; we need completely different techniques. The main idea is the following. Suppose we are looking for an (A, B)-separator S with fractional edge cover number $w < 2$. As the fractional edge cover number is an upper bound on maximum independent set size, any two vertices in S are adjacent; i.e., S induces a clique. The structure of separating cliques is well understood: every graph has a unique decomposition by clique separators [23]. Our algorithm for finding a separator with small fractional edge cover number can be thought of as a generalization of finding clique separators. A tempting way of generalizing this idea for larger w would be to suppose that every separator with fractional edge cover number at most w can be covered by $f(w)$ cliques for some function f. However, this is not true: we might need an unbounded number of cliques (see Example 2.1). Nevertheless, we manage to transform the instance in such a way that it can be assumed that the separator we are looking for can be covered by w cliques. Then we locate these cliques using a combination of brute force, clique separator decompositions, and linear programming.

We finish the paper by proving that it is NP-hard to decide whether the fractional hypertree width of a hypergraph is at most w (Section 5). The hardness result assumes that w is a value given in the input; the much more interesting question of whether the problem is NP-hard for some fixed $w \geq 1$ remains open.

2 Preliminaries

A *hypergraph* is a pair $H = (V(H), E(H))$, consisting of a set $V(H)$ of *vertices* and a set $E(H)$ of subsets of $V(H)$, the *hyperedges* of H. We always assume that hypergraphs have no isolated vertices, that is, for every $v \in V(H)$ there exists at least one $e \in E(H)$ such that $v \in e$. Let $\|H\| := |V(H)| + |E(H)|$, we will express the running time of the algorithms as a function of $\|H\|$.

For a hypergraph H and a set $X \subseteq V(H)$, the *subhypergraph of H induced by X* is the hypergraph $H[X] = (X, \{e \cap X \mid e \in E(H)\})$. We let $H \setminus X = H[V(H) \setminus X]$. The *primal graph* of a hypergraph H is the graph

$$\underline{H} = (V(H), \{\{v, u\} \mid v \neq u, \text{ there exists an } e \in E(H) \text{ such that } \{v, u\} \subseteq e\}).$$

A hypergraph H is *connected* if $\underline{H}$ is connected. A set $C \subseteq V(H)$ is *connected (in H)* if the induced subhypergraph $H[C]$ is connected, and a *connected component* of H is a maximal connected subset of $V(H)$. A sequence of vertices of H is a *path* of H if it is a path of $\underline{H}$. A subset $K \subseteq V(H)$ is a *clique* of H if K induces a clique in $\underline{H}$.

An *edge cover* of a set $S \subseteq V(H)$ is a set $F \subseteq E(H)$ such that for every $v \in S$, there is an $e \in F$ with $v \in e$. The size of the smallest edge cover of S, denoted by $\varrho_H(S)$, is the *edge cover number* of S. A *fractional edge cover* of $S \subseteq V(H)$ is a mapping $\gamma : E(H) \to [0,1]$ such that for every $v \in S$, we have $\sum_{e\in E(H):v\in e}\gamma(e) \geq 1$. The *weight* of the assignment γ is $\text{weight}(\gamma) := \sum_{e\in E(H)}\gamma(e)$. The *fractional edge cover number* of S, denoted by $\varrho^*_H(S)$, is the minimum of weight(γ) taken over every fractional edge cover of S. It is well known that $\varrho^*_H(S) \leq \varrho_H(S) \leq \varrho^*_H(S)(1+\ln|V(H)|)$; in fact, a simple greedy algorithm can be used to find an edge cover of S with size at most $\varrho^*_H(S)(1+\ln|V(H)|)$ (cf. [24]). Note that determining $\varrho_H(S)$ is NP-hard, while $\varrho^*_H(S)$ can be determined in polynomial time using linear programming. We define $\varrho(H)$ and $\varrho^*(H)$ to be $\varrho_H(V(H))$ and $\varrho^*_H(V(H))$, respectively.

EXAMPLE 2.1. For $n \geq 1$, let H_n be the following hypergraph: H_n has a vertex v_S for every subset S of $\{1,\ldots,3n\}$ of cardinality n. Furthermore, for every $i \in \{1,\ldots,3n\}$ the hypergraph H_n has a hyperedge $e_i = \{v_S \mid i \in S\}$. Observe that the fractional edge cover number $\varrho^*(H_n)$ is at most 3, because the mapping ψ that assigns $1/n$ to every hyperedge e_i is a fractional edge cover of weight 3. Actually, it is easy to see that $\varrho^*(H_n) = 3$. On the other hand, the edge cover number cannot be bounded by a constant. Every edge cover has size at least $2n+1$: if $e_{i_1}, \ldots, e_{i_n}$ are n edges *not* present in the edge cover, then the vertex corresponding to the set $\{i_1,\ldots,i_n\}$ is not covered by any edges of the cover. The primal graph of H_n is the complement of the Kneser graph $KG_{3n,n}$. The chromatic number of $KG_{3n,n}$ is known to be $3n-2n+2 = n+2$ [17, 18]. Thus the primal graph H_n cannot be covered by less than $n+2$ cliques. This shows that there is no function $f(w)$ such that every hypergraph H with $\varrho^*(H) \leq w$ can be covered by at most $f(w)$ cliques.

A *tree decomposition* $\mathcal{T}$ of a hypergraph H is a tuple $(T,(B_t)_{t\in V(T)})$, where T is a tree and $(B_t)_{t\in V(T)}$ a family of subsets of $V(H)$ such that for each $e \in E(H)$ there is a node $t \in V(T)$ such that $e \subseteq B_t$, and for each $v \in V(H)$ the set $\{t \in V(T) \mid v \in B_t\}$ is connected in T. The sets B_t are called the *bags* of the decomposition. We denote by $|\mathcal{T}| := |V(T)|$ the number of bags in $\mathcal{T}$. The *width* of a tree decomposition $(T,(B_t)_{t\in V(T)})$ is $\max\{|B_t| \mid t \in V(t)\} - 1$. The *tree width* tw$(H)$ of a hypergraph H is the minimum of the widths of all tree decompositions of H. It is easy to see that tw$(H) = $ tw$(\underline{H})$ for all H.

The *generalized hypertree width* of a decomposition $(T,(B_t)_{t\in V(T)})$ is $\max\{\varrho_H(B_t) \mid t \in V(t)\}$ and the *generalized hypertree width* of a hypergraph H, denoted by ghw(H), is the minimum of the generalized hypertree widths of all tree decompositions of H. *Fractional hypertree width* of a tree decomposition and of a hypergraph is defined analogously, by having $\varrho^*_H(B_t)$ instead of $\varrho_H(B_t)$ in the definition. We denote by fhw(H) the fractional hypertree width of H.

3 Finding approximate separators

Let $A, B \subseteq V(H)$ be two sets of vertices. An (A,B)-*separator* is a set $S \subseteq V(H)$ such that there is no path connecting a vertex of $A \setminus S$ with a vertex of $B \setminus S$ in the hypergraph $H \setminus S$. In particular, such an S has to contain every vertex of $A \cap B$. The aim of this section is to give an approximation algorithm for the problem of finding an (A,B)-separator with minimum fractional edge cover number.

We say that two nonadjacent vertices u, v of H are w-*attached* for some $w \geq 1$ if $\varrho^*_H(N(v)\cap N(u)) > w$ (here $N(v)$ is the set of neighbors of v, not including v itself). If S is an (A,B)-separator with $\varrho^*_H(S) \leq w$ covering neither u nor v, and u, v are w-attached, then u and v are in the same connected component of $H \setminus S$. This means that S remains an (A,B)-separator even if we add an edge between u and v. Thus adding edges between w-attached vertices does not change the problem significantly. More precisely, the following lemma shows that the we can reduce the problem to a situation where nonadjacent vertices are not w-attached. This property of the hypergraph will play an important role in the algorithm.

LEMMA 3.1. *Let H be a hypergraph, $A, B \subseteq V(H)$ sets of vertices, and $w \geq 1$ a rational number. We can construct in time polynomial in $\|H\|$ a hypergraph H^+ on the same set of vertices such that*

1. *If vertices u and v are not adjacent in H^+, then they are not w-attached.*
2. *If S is an (A,B)-separator in H with $\varrho^*_H(S) \leq w$, then S is an (A,B)-separator in H^+ with $\varrho^*_{H^+}(S) \leq w$.*
3. *If S is an (A,B)-separator in H^+, then S is an (A,B)-separator in H with $\varrho^*_H(S) \leq 2\varrho^*_{H^+}(S)$.*

Proof. We construct a sequence of hypergraphs. Let $H_0 = H$. Let (u,v) be an arbitrary pair of nonadjacent vertices that are w-attached in H_{i-1}. Hypergraph H_i is the same as H_{i-1} with an extra edge $\{u,v\}$. If there is no such pair (u,v) in H_{i-1}, then we stop the construction of the sequence. It is clear that the sequence has polynomial length (as at most $O(|V(H)|^2)$ new edges can be added) and constructing H_i from H_{i-1} can be done in polynomial time. Let $H^+ = H_k$

be the last hypergraph in the sequence. Statement 1 is immediate from the way the sequence is constructed.

To prove Statement 2, suppose that S is an (A, B)-separator in $H = H_0$. Since the edges of H are a subset of the edges of H^+, we have $\varrho^*_{H^+}(S) \le \varrho^*_H(S) \le w$. We prove by induction that S is an (A, B)-separator in every H_i. Suppose that this is true for H_{i-1}, but there is a path P from a vertex of A to a vertex of B in $H_i \setminus S$. Let $e_i = u_i v_i$ be the edge that was added to H_{i-1} to obtain H_i. If P does not use e_i, then P is also a path in H_{i-1}, contradicting the induction hypothesis that S is an (A, B)-separator in H_{i-1}. Thus $P = P_1 u_i v_i P_2$ for some subpaths P_1 and P_2. By the definition of e_i,

$$\varrho^*_H(N(v_i) \cap N(u_i)) \ge \varrho^*_{H_{i-1}}(N(v_i) \cap N(u_i)) > w \ge \varrho^*_H(S),$$

which means that there is a vertex $q \in (N(v_i) \cap N(u_i)) \setminus S$. The walk $P_1 u_i q v_i P_2$ connects a vertex of A and a vertex of B in $H_{i-1} \setminus S$, contradicting the induction hypothesis.

To prove Statement 3, observe first that the edges of H are a subset of the edges of H^+, thus if S is an (A, B)-separator in H^+, then it is an (A, B)-separator in H as well. Consider a fractional edge cover γ of S in H^+ with weight$(\gamma) = w'$. Suppose that $\gamma(e) = x$ for an edge $e = \{u, v\}$ not present in H. In this case, we set the weight of this edge to 0, and increase by x the weight of two edges: an arbitrary edge $e_u \in E(H)$ that contains u and an arbitrary edge $e_v \in E(H)$ that contains v (such edges exist, since we assumed that there are no isolated vertices in the hypergraph). It is clear that the resulting weight assignment is also a fractional edge cover. We repeat this step until the weight assignment is 0 on every edge not present in H. It is easy to see that the weight of the assignment increases to at most $2w'$, thus $\varrho^*_H(S) \le 2\varrho^*_{H^+}(S)$. $\square$

The following result follows from the fact that a decomposition of a graph by clique separators can be found in polynomial time [25, 23]. For the convenience of the reader, we give here a self-contained proof of the main idea in the form we use.

LEMMA 3.2. *Given a graph G, it is possible to construct in time polynomial in $\|G\|$ a set $\mathcal{C}$ of at most $|V(G)|$ connected subsets such that*

1. *if K is a clique of G, then $K \subseteq C$ for some $C \in \mathcal{C}$, and*
2. *if K is a clique of G and $C \in \mathcal{C}$, then $C \setminus K$ is contained in a connected component of $G \setminus K$.*

Proof. We construct a sequence of graphs as follows. Let $G_0 = G$. Suppose that G_{i-1} has an induced cycle H of length at least 4; let v_i, u_i be two nonadjacent vertices of H. We define G_i to be the same as G_{i-1}, with an extra edge $e_i = v_i u_i$. If G_{i-1} has no such cycle H (i.e., G_{i-1} is a chordal graph), then we stop the construction of the sequence. Let G_k be the last graph in the sequence. Let $\mathcal{C}$ be the set of inclusionwise maximal cliques of G_k. It is well known that chordal graph G_k has at most $|V(G_k)| = |V(G)|$ maximal cliques.

Every clique of G is a clique of G_k, thus Statement 1 is clear from the definition of $\mathcal{C}$. To prove Statement 2, for every $C \in \mathcal{C}$ and clique K of G, we show that $C \setminus K$ is contained in a connected component of $G_i \setminus K$ for every $1 \le i \le k$. This is clear for G_k, as C is a clique in G_k. Suppose that $C \setminus K$ is in a connected component of $G_i \setminus K$ but $a, b \in C \setminus K$ are in different connected components of $G_{i-1} \setminus K$. Let P be a path from a to b in $G_i \setminus K$. Path P has to go through the edge $e_i = v_i u_i$ used in the definition of G_i, otherwise it would be a path in $G_{i-1} \setminus K$ as well. Thus the path P can be written as $P = aP_1 v_i u_i P_2 b$. There is a induced cycle H in G_{i-1} that contains v_i and u_i. Since $v_i, u_i \notin K$ and $H \setminus K$ is connected (as K is a clique), there is a path R in $G_{i-1} \setminus K$ that connects v_i and u_i. Now $aP_1 v_i R u_i P_2 b$ is a walk from a to b in $G_{i-1} \setminus K$, a contradiction. $\square$

LEMMA 3.3. *Let H be a hypergraph, $A, B \subseteq V(H)$ two sets of vertices, and $w \ge 1$ a rational number. There is an algorithm that, in time $\|H\|^{O(w)}$, either*

- *correctly concludes that there is no (A, B)-separator S with $\varrho^*_H(S) \le w$, or*
- *produces an (A, B)-separator S' with $\varrho^*_H(S') \le w^3 + 4w$.*

Proof. The algorithm first constructs the hypergraph H^+ of Lemma 3.1 and then tries to find an (A, B)-separator in H^+. By Lemma 3.1(2), if H has an (A, B)-separator S with $\varrho^*_H(S) \le w$, then S is an (A, B)-separator in H^+ as well and $\varrho^*_{H^+}(S) \le w$. In this case, our algorithm will be able to find an (A, B)-separator S' in H^+ with $\varrho^*_{H^+}(S') \le w^3/2 + 2w$. By Lemma 3.1(3), such an S' is an (A, B)-separator in H with $\varrho^*_H(S') \le w^3 + 4w$.

Suppose that there is an (A, B)-separator S in H^+ with $\varrho^*_{H^+}(S) \le w$. In the rest of the proof, we show how to find the required separator S' if we know a maximum independent set I_S of S. Since the fractional edge cover number of S is at most w, the size of I_S is also at most w. Thus trying all possible sets I_S adds a factor of $\|H^+\|^{O(w)} = \|H\|^{O(w)}$ to the running time.

Suppose that $I_S = \{v_1, \dots, v_k\}$ (for some $k \le w$) is a maximum independent set of S. By the definition of H^+, we have $\varrho^*_{H^+}(N(v_i) \cap N(v_j)) \le w$ for every $1 \le i < j \le k$. Thus $X = \bigcup_{1 \le i < j \le k}(N(v_i) \cap N(v_j))$ has fractional edge cover number at most $\binom{k}{2} w \le w^3/2$. In the rest of the algorithm, we try to find a set Y with $\varrho^*_{H^+}(Y) \le 2w$ such that $S' := X \cup Y$ is an (A, B)-separator in H^+.

Denote by $N(v_i)$ the neighbors of v_i in H^+. Let $N_i = (N(v_i) \cup \{v_i\}) \setminus X$ for $i = 1, \dots, k$. Let us note first that $N_i \cap N_j = \emptyset$ if $i \neq j$: vertices v_i and v_j are not adjacent and every vertex of $N(v_i) \cap N(v_j)$ is in X. Since $v_1, \dots, v_k$ is a maximum independent set of S, each vertex of $S \setminus X$ is in one of the N_i's. Observe that $N_i \cap S$ is not empty, since it contains v_i (here we use that v_i cannot be in X, since it is not adjacent to any other v_j). Furthermore, for every $1 \le i \le k$, $N_i \cap S$ is a clique of N_i. To see this, suppose that $v'_i, v''_i \in N_i \cap S$ are nonadjacent vertices. Vertices v'_i and v''_i cannot be adjacent to any v_j with $i \neq j$: that would imply that they are in $N(v_i) \cap N(v_j) \subseteq X$. Thus replacing v_i in I_S with v'_i and v''_i would give a strictly larger independent set, contradicting the maximality of I_S.

Let $\underline{H}$ be the primal graph of H^+. For every $1 \le i \le k$, let $C_{i,1}, \dots, C_{i,c_i}$ be the connected sets given by Lemma 3.2 for the graph $\underline{H}[N_i]$. By the definition of these sets, for every $1 \le i \le k$ there is a value $1 \le d_i \le c_i$ such that the clique $N_i \cap S$ is fully contained in C_{i,d_i}. Furthermore, the connected set $C_{i,d_i} \setminus (N_i \cap S)$ is contained in a connected component of $\underline{H}[N_i \setminus (N_i \cap S)]$, which implies that $C_{i,d_i} \setminus S$ is contained in a connected component of $\underline{H} \setminus S$. Thus either every vertex of $C_{i,d_i} \setminus S$ is reachable from A in $\underline{H} \setminus S$, or none of these vertices are reachable. Let us define $a_i = 1$ in the first case and $a_i = 0$ in the second case.

We show that if the values d_i, a_i ($1 \le i \le k$) corresponding to S are known, then the required separator S' can be found. Thus we have to try all possibilities for these values, which adds a factor of $|V(H)|^{O(w)} \cdot 2^{O(w)}$ to the running time.

Suppose that the values of d_i, a_i are given. Let $Z := X \cup \bigcup_{i=1}^{k} C_{i,d_i}$; note that $S \subseteq Z$. We say that a vertex $u \in C_{i,d_i}$ is a *bad vertex* if

- $a_i = 0$ and there is a path P_a from A to u with $P_a \cap Z = \{u\}$, or
- $a_i = 1$ and there is a path P_b from B to u with $P_b \cap Z = \{u\}$.

(It is possible that P_a or P_b consists of only the vertex u; in particular, if $u \in A \cap B$, then u is always a bad vertex.) Observe that S contains every bad vertex u. Indeed, if $u \notin S$ and there is a path P_a as above, then $S \cap P_a = \emptyset$ (since $S \subseteq Z$), thus u is reachable from A, contradicting $a_i = 0$. On the other hand, if $u \notin S$ and there is a path P_b, then u is reachable from B, but $a_i = 1$ implies that it is also reachable from A, contradicting the fact that S is an (A, B)-separator.

A pair $u \in C_{i,d_i}$ and $v \in C_{j,d_j}$ is a *bad pair* if

- there is a path P from u to v with $P \cap Z = \{u, v\}$ and $a_i \neq a_j$.

In this case, S has to contain at least one of u and v: otherwise $P \cap S = \emptyset$ would mean that u and v are in the same connected component of $H^+ \setminus S$, implying $a_i = a_j$. Thus every fractional edge cover of S is a solution of the following linear program:

$$\min \sum_{e \in E(H^+)} x_e$$

$$\sum_{\substack{e \in E(H^+)\\ v \in e}} x_e \ge 1 \qquad \forall v \in Z, v \text{ is a bad vertex}$$

$$\sum_{\substack{e \in E(H^+)\\ u \in e}} x_e + \sum_{\substack{e \in E(H^+)\\ v \in e}} x_e \ge 1 \quad \forall u, v \in Z, u, v \text{ is a bad pair}$$

Therefore, the optimum of the linear program is at most w. Let $(x_e)_{e \in E(H^+)}$ be a solution of the linear program with cost at most w. Let Y contain those vertices v for which $\sum_{e \in E(H^+): v \in e} x_e \ge 1/2$; clearly, $\varrho^*_{H^+}(Y) \le 2w$. Thus defining $S' := X \cup Y$ gives a set with $\varrho^*_{H^+}(Y) \le w^3/2 + 2w$. Observe that the linear program ensures that S' contains every bad vertex and at least one vertex from each bad pair.

We claim that S' is an (A, B)-separator in H^+. Suppose that there is a path P from $a \in A$ to $b \in B$ in $H^+ \setminus S'$. This path contains at least one vertex of S (since S is an (A, B)-separator), hence it contains at least one vertex of Z. Let $p_1, \dots, p_r$ be the vertices of $P \cap Z$, ordered as the path is traversed from a to b. Since these vertices cannot be in $X \subseteq S'$, they are in $\bigcup_{i=1}^{k} C_{i,d_i}$. Suppose first that p_1 is not reachable from A in $H^+ \setminus S$. This means that if N_i is the set that contains p_1, then $a_i = 0$. It follows that p_1 is a bad vertex (because of the subpath of P that connects a with p_1), hence $p_1 \in S'$, a contradiction. Let $1 \le \ell \le r$ be the largest value such that p_ℓ is reachable from A in $H^+ \setminus S$ and suppose that p_ℓ is in N_i. If $\ell = r$, then p_ℓ is a bad vertex (because of $a_i = 1$ and the subpath of P connecting p_ℓ and b), again a contradiction. Finally, if $\ell < r$, then let N_j be the set that contains $p_{\ell+1}$. The maximality of ℓ implies $a_i = 1$ and $a_j = 0$. Therefore, $p_\ell, p_{\ell+1}$ is a bad pair (because of the subpath of P connecting these two vertices), and S' contains at least one of these vertices, a contradiction. Thus S' is an (A, B)-separator in H^+ with $\varrho^*_{H^+} \le w^3/2 + 2w$.

In summary, the algorithm performs the following steps:

1. Construct the hypergraph H^+ (Lemma 3.1).
2. Guess the independent set I_S.
3. Construct the set X and define the sets N_i.
4. Construct the sets $C_{i,j}$ (Lemma 3.2).
5. Guess the values d_i, a_i.
6. Construct Y using an optimum solution of the linear program.
7. Check if $S' := X \cup Y$ is an (A, B)-separator.

As discussed above, if there is an (A,B)-separator S with $\varrho^*_H(S) \le w$, then it is possible to choose I_S and the values d_i, a_i such that the separator S' computed by the algorithm is an (A,B)-separator with $\varrho^*_H(S') \le w^3 + 4w$. Thus if we try all possible $\|H\|^{O(w)} \cdot \|H\|^{O(w)} \cdot 2^{O(w)}$ guesses, then we will find such a separator S' in this case. On the other hand, if none of the guesses results in the required separator S', then we can correctly conclude that there is a no (A,B)-separator S in H with $\varrho^*_H(S) \le w$. The running time of each step (except the guesses) is polynomial, thus the total running time is $\|H\|^{O(w)}$. □

In the tree decomposition algorithm of Section 4, we have to find a balanced separation of a set W: we need a partition (A,B) of W such that (1) $\varrho^*_H(A)$, $\varrho^*_H(B)$ are not too large and (2) there is an (A,B)-separator S such that $\varrho^*_H(S)$ is not too large. As we shall see, it follows from the results of [14] that such a balanced separation always exists if H has bounded fractional hypertree width. If we want to find such a separation algorithmically, then the main problem is how to find the partition (A,B) of W: if (A,B) is given, then Lemma 3.3 can be used to find an (A,B)-separator whose fractional edge cover number is bounded. Trying all possible partitions of W is not feasible. Fortunately, for the applications in Lemma 3.4, we can assume that $\varrho^*_H(W)$ is bounded. Instead of trying all possible partitions of W, it turns out that it is sufficient to try all possible partitions of an edge cover of W.

LEMMA 3.4. *Let H be a hypergraph with fractional hypertree width at most w and let $W \subseteq V(H)$ be a subset of vertices with $\varrho^*_H(W) \le k$. It is possible to find in time $\|H\|^{O(w+k)}$ a partition (A,B) of W and an (A,B)-separator S with $\varrho^*_H(S) \le w^3 + 4w$ such that $\varrho^*_H(A), \varrho^*_H(B) \le \frac{2}{3}k + w$.*

Proof. Since the fractional edge cover number of W is at most k, the greedy algorithm finds an edge cover $F \subseteq E(H)$ of W with $|F| = O(k \log |V(H)|)$. Our algorithm tries every partition (F_A, F_B) of F, defines $A := W \cap \bigcup F_A$ and $B := W \setminus A$, and checks whether the algorithm of Lemma 3.3 produces an (A,B)-separator S with $\varrho^*_H(S) \le w^3 + 4w$. We show that if H has fractional hypertree width at most w, then at least one partition (F_A, F_B) results in a partition (A,B) and a separator S satisfying the conditions. Trying every possible partition (F_A, F_B) means trying $2^{O(k \log |V(H)|)} = \|H\|^{O(k)}$ possibilities and the algorithm of Lemma 3.3 needs $\|H\|^{O(w)}$ time. Thus the total running time of the algorithm is $\|H\|^{O(k+w)}$.

By [14, Theorem 11, Lemma 12], there is a set S_0 with $\varrho^*_H(S_0) \le w$ such that $\varrho^*_H(C \cap W) \le k/2$ for every connected component C of $H \setminus S_0$; let $C_1, \dots, C_d$ be these connected components. Define $W_i := W \cap C_i$ and suppose that the connected components are ordered such that $\varrho^*_H(W_i) \ge \varrho^*_H(W_j)$ if $i < j$. Let ℓ be the smallest value such that $\varrho^*_H(\bigcup_{i=1}^{\ell} W_i) \ge \frac{1}{3}k$. Observe that $\varrho^*_H(\bigcup_{i=1}^{\ell} W_i) \le \frac{2}{3}k$: if $\varrho^*_H(W_1) \ge \frac{1}{3}k$, then $\ell = 1$ and $\varrho^*_H(W_1) \le k/2$; if $\varrho^*_H(W_1) < \frac{1}{3}k$, then $\ell > 1$ and $\varrho^*_H(\bigcup_{i=1}^{\ell} W_i) \le \varrho^*_H(\bigcup_{i=1}^{\ell-1} W_i) + \varrho^*_H(W_\ell) < \frac{1}{3}k + \frac{1}{3}k$. Furthermore, $\varrho^*_H(\bigcup_{i=1}^{d} W_i) = \varrho^*_H(\bigcup_{i=1}^{\ell} W_i) + \varrho^*_H(\bigcup_{i=\ell+1}^{d} W_i)$, since there is no edge that intersects more than one W_i. From $\varrho^*_H(\bigcup_{i=1}^{d} W_i) \le \varrho^*_H(W) \le k$, we have $\varrho^*_H(\bigcup_{i=\ell+1}^{d} W_i) \le \frac{2}{3}k$.

Let F_A be the edges of F fully contained in $S_0 \cup \bigcup_{i=1}^{\ell} C_i$ and let F_B be the edges of F intersecting $\bigcup_{i=\ell+1}^{d} C_i$. Observe that (F_A, F_B) is a partition of F. Let $A := W \cap \bigcup F_A$ and $B := W \setminus A$ be defined as in the algorithm. Since $A \subseteq S_0 \cup (W \cap \bigcup_{i=1}^{\ell} C_i)$, we have $\varrho^*_H(A) \le \varrho^*_H(S_0) + \varrho^*_H(\bigcup_{i=1}^{\ell} W_i) \le w + \frac{2}{3}k$. Similarly, $\varrho^*_H(B) \le w + \frac{2}{3}k$. Observe that S_0 is an (A,B)-separator with $\varrho^*_H(S_0) \le w$, thus the algorithm of Lemma 3.3 produces an (A,B)-separator S with $\varrho^*_H(S) \le w^3 + 4w$. Therefore, when the algorithm considers this particular partition (F_A, F_B), then it finds the required partition (A,B) and separator S. □

4 Finding approximate tree decompositions

We prove the main result of the paper in this section: it is possible to approximate fractional hypertree width in a sense that is suitable for the applications. That is, if a class $\mathcal{H}$ of hypergraphs has bounded fractional hypertree width, then there is a polynomial time algorithm producing a tree decomposition with bounded fractional hypertree width for any hypergraph in $\mathcal{H}$. The algorithm uses the balanced separation algorithm of Lemma 3.4.

THEOREM 4.1. *Given a hypergraph H and a rational number $w \ge 1$, it is possible in time $\|H\|^{O(w^3)}$ to either*

- *compute a fractional hypertree decomposition of H with width at most $7w^3 + 31w + 7$, or*
- *correctly conclude that* $\mathrm{fhw}(H) > w$.

Proof. We present an algorithm for a more general problem:

> Given a hypergraph H with $\mathrm{fhw}(H) \le w$ and a set W with $\varrho^*_H(W) \le 6w^3 + 27w + 6$, find a fractional hypertree decomposition $\mathcal{T}$ of width at most $7w^3 + 31w + 7$ such that some bag B of $\mathcal{T}$ contains the set W.

(Note that this algorithm implies the existenece of the algorithm required by the theorem: if this algorithm is applied to a hyerpgraph H with $\mathrm{fhw}(H) > w$, then either it produces a fractional hypertree decomposition of H with the required width or if the output is something else, then we can correctly conclude that $\mathrm{fhw}(H) > w$.) If $\varrho^*(H) \le 7w^3 + 31w + 7$, then we are done: a tree decomposition consisting of a single bag $B = V(H)$ is sufficient. Thus we can assume that $\varrho^*(H) \geq 7w^3 + 31w + 7$. By adding arbitrary vertices to W one by one, we can extend W such that $6w^3 + 27w + 6 \leq \varrho^*_H(W) < 6w^3 + 27w + 7$. Let us use the algorithm of Lemma 3.4 to find a partition (A, B) of W and an (A, B)-separator S. A connected component of $H \setminus S$ cannot intersect both A and B. Let V_1 be the union of S and all the connected components intersecting A; let V_2 be the union of S and the connected components not intersecting A. Let H_1 (resp., H_2) be the subhypergraph of H induced by V_1 (resp., V_2).

First we verify that H_1 and H_2 are proper subhypergraphs of H; in fact, their fractional edge cover number is strictly less than $\varrho^*(H)$. Since $\varrho^*_H(W) \leq \varrho^*_H(W \cap V_1) + \varrho^*_H(W \setminus V_1)$ and $\varrho^*_H(W \cap V_1) \leq \varrho^*_H(A) + \varrho^*_H(S)$, we have

$$(4.1) \quad \varrho^*_H(W \setminus V_1) \geq \varrho^*_H(W) - \frac{2}{3}\varrho^*_H(W) - w - \varrho^*_H(S) \geq w^3 + 4w + 2.$$

Consider a fractional edge cover γ of H with weight $\varrho^*(H)$. Let γ_S be a fractional edge cover of S with weight $\varrho^*_H(S)$. Let us define

$$\gamma'(e) = \begin{cases} \gamma(e) & \text{if } e \cap (W \setminus V_1) = \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

Observe that weight$(\gamma') \leq$ weight$(\gamma) - (w^3 + 4w + 2)$, since by (4.1), γ has to assign weight at least $w^3 + 4w + 2$ to the edges intersecting $W \setminus V_1$. Now $\gamma' + \gamma_S$ is an edge cover of V_1 (since edges intersecting $W \setminus V_1$ cannot intersect $V_1 \setminus S$), thus $\varrho^*(H_1) \leq$ weight$(\gamma')+$weight$(\gamma_S) \leq \varrho^*(H) - (w^3 + 4w + 2) + \varrho^*_H(S) \leq \varrho^*(H) - 2$. A similar argument shows $\varrho^*(H_2) \leq \varrho^*(H) - 2$.

Let $W_1 := A \cup S$ and $W_2 := B \cup S$; we have $\varrho^*_H(W_1), \varrho^*_H(W_2) \leq \frac{2}{3}\varrho^*_H(W) + w + \varrho^*_H(S) < 6w^3 + 27w + 6$. Since H_1 and H_2 are strictly smaller than H, we can use the algorithm recursively to obtain a tree decomposition $\mathcal{T}_1$ of H_1 where W_1 is contained in some bag B_1, and a tree decomposition $\mathcal{T}_2$ of H_2 where W_2 is contained in some bag B_2. We connect these two tree decomposition by introducing a new bag $B_0 := W \cup S$ that is connected to B_1 and B_2; note that $\varrho^*_H(B_0) \leq 7w^3 + 31w + 7$. It is easy to see that the resulting tree decomposition $\mathcal{T}$ is a proper tree decomposition of H and the bag B_0 fully contains W.

Let us estimate the running time of the algorithm. If $\varrho^*(H) \leq 7w^3 + 31w + 7$, then the algorithm constructs only a single bag and does not recurse. We prove by induction that if $\varrho^*(H) \geq 7w^3 + 31w + 7$, then the algorithm constructs a tree decomposition with at most $\varrho^*(H) - 2w^3 - 8w - 1$ bags. As the time spent constructing a bag is $\|H\|^{O(w^3)}$, this proves that the running time is $\|H\|^{O(w^3)}$.

First we show that

$$(4.2) \qquad \varrho^*_H(V_1) + \varrho^*_H(V_2) \leq \varrho^*(H) + 2w^3 + 8w.$$

To see this, consider a fractional edge cover γ of H with weight $\varrho^*(H)$ and let γ_S be a fractional edge cover of S with weight $w^3 + 4w$. Let us define

$$\gamma_1(e) = \begin{cases} \gamma(e) & \text{if } e \not\subseteq V_2 \\ 0 & \text{otherwise} \end{cases} \text{ and } \gamma_2(e) = \begin{cases} \gamma(e) & \text{if } e \not\subseteq V_1 \\ 0 & \text{otherwise.} \end{cases}$$

Since every edge is fully contained in either V_1 or V_2, we have weight(γ_1) + weight$(\gamma_2) \leq$ weight(γ). Furthermore, $\gamma_1 + \gamma_S$ is an edge cover of V_1, and $\gamma_2 + \gamma_S$ is an edge cover of V_2. Now (4.2) follows from weight$(\gamma_S) \leq w^3 + 4w$. Subtracting $4w^3 + 16w + 2$ from both sides of (4.2), we get

$$(4.3) \quad (\varrho^*(H_1) - 2w^3 - 8w - 1) + (\varrho^*(H_2) - 2w^3 - 8w - 1) \leq (\varrho^*(H) - 2w^3 - 8w - 1) - 1$$

Suppose that hypergraph H with $\varrho^*(H) > 7w^3+31w+7$ is decomposed into H_1 and H_2. The algorithm constructs a tree decomposition $\mathcal{T}$ that is obtained by joining the tree decompositions $\mathcal{T}_1$ and $\mathcal{T}_2$ with a new bag. Thus $|\mathcal{T}| = |\mathcal{T}_1|+|\mathcal{T}_2|+1$. We have to consider different cases depending on how $\varrho^*(H_1)$, $\varrho^*(H_2)$ compare with $7w^3 + 31w + 7$. If $\varrho^*(H_1), \varrho^*(H_2) > 7w^3 + 31w + 7$, then the induction hypothesis and (4.3) shows $|\mathcal{T}| \leq \varrho^*(H) - 2w^3 - 8w - 1$. If $\varrho^*(H_1), \varrho^*(H_2) \leq 7w^3 + 31w + 7$, then $\mathcal{T}$ consists of only 3 bags. Since $\varrho^*(H) - 2w^3 - 8w - 1 \geq 5w^3 + 23w + 6 \geq 3$, the induction statement holds in this case as well. Suppose now that $\varrho^*(H_1) > 7w^3 + 31w + 7$ and $\varrho^*(H_2) \leq 7w^3 + 31w + 7$. In this case, $|\mathcal{T}| = |\mathcal{T}_1| + 2$. Now $|\mathcal{T}| \leq \varrho^*(H) - 2w^3 - 8w - 1$ follows from the induction hypothesis on H_1 and $\varrho^*(H_1) \leq \varrho^*(H)-2$ proved earlier. The case when $\varrho^*(H_1) \leq 7w^3 + 31w + 7$ and $\varrho^*(H_2) > 7w^3 + 31w + 7$ can be proved similarly. $\square$

5 Hardness result

Gottlob et al. [8] have shown that, given a hypergraph H and an integer k, it is NP-hard to decide if ghw$(H) \leq k$. The proof is a very simple reduction from SET COVER. This proof cannot be adapted to prove hardness for fractional hypertree width, since the fractional version of SET COVER is polynomial-time solvable. Here we prove the hardness of fractional hypertree width using the fact that given a graph G and an integer k, it is NP-hard to decide if the tree width of G is at most k [3]. Note that for every fixed k, it can be checked in linear time whether the tree width is at most k [2], thus tree width is hard only if k is part of the input. Consequently, our hardness result for fractional hypertree width assumes that the bound w is given in the input. This means that the hardness result does not rule out the possibility that for every fixed $w \geq 1$, there is a polynomial-time algorithm for deciding $\mathrm{fhw}(H) \leq w$ (and for constructing the corresponding decomposition). It remains an interesting open question whether the approximation algorithm presented in this paper can be replaced by an optimal polynomial-time algorithm or the problem is NP-hard already for some fixed $w \geq 1$. Note that for generalized hypertree with, Gottlob et al. [10] gave a (much more involved) proof that deciding $\mathrm{ghw}(H) \leq 3$ is NP-hard.

THEOREM 5.1. *Given a hypergraph H and rational number $w \geq 1$, it is NP-hard to decide whether* $\mathrm{fhw}(H) \leq w$.

Proof. Given a graph G and an integer k, we construct a hypergraph H such that $\mathrm{tw}(G) \leq k$ if and only if $\mathrm{fhw}(H) \leq k+1$. Let $v_1, \dots, v_n$ be the vertices of G. The hypergraph H is obtained by adding new vertices and edges to G. Let $a_{i,j}$ ($1 \leq i \leq k+1$, $1 \leq j \leq 3$) be new vertices and let A be the set of these $3(k+1)$ vertices. For every $1 \leq x \leq n$, we add $k+1$ new edges $e_{x,i} = \{v_x, a_{i,1}, a_{i,2}, a_{i,3}\}$. Finally, for every pair $a', a'' \in A$, we add an edge $\{a', a''\}$. This completes the description of H.

Suppose that $(T, (B_t)_{t \in V(T)})$ is a width k tree decomposition of G. For every $t \in V(T)$, let $B'_t = B_t \cup A$. It is easy to see that $(T, (B_t)_{t \in V(T)})$ is a tree decomposition of H. Furthermore, $\varrho^*_H(B'_t) \leq k+1$ for every $t \in V(T)$: if $B_t = \{v_{x_1}, \dots, v_{x_{k+1}}\}$, then the edges $e_{x_1,1}$, $e_{x_2,2}$, $\dots$, $e_{x_{k+1},k+1}$ form an edge cover of $B_t \cup A$.

Suppose now that $(T, (B'_t)_{t \in V(T)})$ is a tree decomposition of H with fractional hypertree width at most $k+1$. First we show that it can be assumed that every B'_t contains A. It is well known that every clique is fully contained in some bag of the decomposition. Since A is a clique, there is at least one bag containing A and, by the properties of the tree decomposition, the bags containing A form a connected subtree T_0 of T. We show that if we replace T with T_0, then it remains a tree decomposition of H. To see that every vertex $v \notin A$ of H appears in a bag of T_0, observe that $A \cup \{v\}$ is a clique, thus there is a bag of T (and hence of T_0) that fully contains $A \cup \{v\}$. Similarly, if $u, v \notin A$ are neighbors, then $A \cup \{u, v\}$ is a clique, and it follows that T_0 has a bag containing both u and v.

Therefore, we can assume that $A \subseteq B'_t$ for every $t \in V(T)$. Let $B_t := B'_t \setminus A$. It is clear that $(T, (B_t)_{t \in V(T)})$ is a tree decomposition of G. Let us show that $|B_t| \leq k+1$ for every $t \in V(T)$. Let γ be a fractional edge cover of B'_t with $\mathrm{weight}(\gamma) \leq k+1$. Denote by $\gamma(a_{i,j}) := \sum_{e \in E(H): a_{i,j} \in e} \gamma(e)$ the weight assigned to the edges containing $a_{i,j}$. As γ is a fractional edge cover of A, the sum $\sum_{i=1}^{k+1} \sum_{j=1}^{3} \gamma(a_{i,j})$ is at least $3(k+1)$. Each edge $e_{x,i}$ contributes to 3 terms of this sum, while every other edge contributes to at most 2 terms. Since the total weight of the edges is $k+1$, this is only possible if γ is nonzero only on the edges of the form $e_{x,i}$. Since each such edge covers only one vertex outside A, the bag B'_t can contain at most $k+1$ vertices outside A, proving $|B_t| \leq k+1$. $\square$

6 Conclusions

The algorithm presented in the paper shows that if $\mathcal{H}$ is a class of hypergraphs with bounded fractional hypertree width, then there is a polynomial-time algorithm that can produce a tree decomposition with bounded hypertree width for each member of $\mathcal{H}$. It follows that CSP instances where the constraint structure has bounded fractional hypertree width are polynomial-time solvable; in fact, this condition is the strictly most general known tractability criterion. It remains an important open question whether there are further tractable cases not covered by bounded fractional hypertree width. As our algorithm computes only an approximately optimal tree decomposition, another open question is whether it can be made an exact algorithm, i.e., $7w^3 + 31w + 7$ in Theorem 4.1 can be replaced with w. We expect that this turns out to be NP-hard, similarly as in the case of generalized hypertree width [10].

References

[1] I. Adler, G. Gottlob, and M. Grohe. Hypertree width and related hypergraph invariants. *European J. Combin.*, 28(8):2167–2181, 2007.

[2] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[3] H. L. Bodlaender, J. R. Gilbert, H. Hafsteinsson, and T. Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *J. Algorithms*, 18(2):238–255, 1995.

[4] F. R. K. Chung, R. L. Graham, P. Frankl, and J. B. Shearer. Some intersection theorems for ordered sets and graphs. *J. Combin. Theory Ser. A*, 43(1):23–37, 1986.

[5] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1999.

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

[7] E. C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proc. of AAAI-90*, pages 4–9, Boston, MA, 1990.

[8] G. Gottlob, M. Grohe, N. Musliu, M. Samer, and F. Scarcello. Hypertree decompositions: structure, algorithms, and applications. In *Graph-theoretic concepts in computer science*, volume 3787 of *Lecture Notes in Comput. Sci.*, pages 1–15. Springer, Berlin, 2005.

[9] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences*, 64:579–627, 2002.

[10] G. Gottlob, Z. Miklós, and T. Schwentick. Generalized hypertree decompositions: NP-hardness and tractable variants. In *PODS '07: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 13–22, New York, NY, USA, 2007. ACM.

[11] G. Gottlob and S. Szeider. Fixed-parameter algorithms for artificial intelligence, constraint satisfaction and database problems. *The Computer Journal*, 51(3):303–325, 2008.

[12] M. Grohe. The structure of tractable constraint satisfaction problems. In *MFCS 2006*, pages 58–72, 2006.

[13] M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1):1, 2007.

[14] M. Grohe and D. Marx. Constraint solving via fractional edge covers. In *SODA '06: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 289–298, New York, NY, USA, 2006. ACM Press.

[15] M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? In *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 657–666, New York, NY, USA, 2001. ACM Press.

[16] P. G. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. *J. Comput. Syst. Sci.*, 61(2):302–332, 2000.

[17] L. Lovász. Kneser's conjecture, chromatic number, and homotopy. *J. Combin. Theory Ser. A*, 25(3):319–324, 1978.

[18] J. Matoušek. A combinatorial proof of Kneser's conjecture. *Combinatorica*, 24(1):163–170, 2004.

[19] S. Oum. Approximating rank-width and clique-width quickly. In *Proceedings of the 31st International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 49–58, 2005.

[20] S. Oum and P. Seymour. Testing branch-width. *J. Combin. Theory Ser. B*, 97(3):385–393, 2007.

[21] S.-i. Oum and P. Seymour. Approximating clique-width and branch-width. *J. Combin. Theory Ser. B*, 96(4):514–528, 2006.

[22] T. J. Schaefer. The complexity of satisfiability problems. In *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing (San Diego, Calif., 1978)*, pages 216–226. ACM, New York, 1978.

[23] R. E. Tarjan. Decomposition by clique separators. *Discrete Math.*, 55(2):221–232, 1985.

[24] V. Vazirani. *Approximation algorithms*. Springer-Verlag, 2004.

[25] S. H. Whitesides. An algorithm for finding clique cut-sets. *Inform. Process. Lett.*, 12(1):31–32, 1981.