An FPT 2-Approximation for Tree-Cut Decomposition

Eun Jung Kim, Sang-Il Oum, Christophe Paul, Ignasi Sau, Dimitrios M. Thilikos

An FPT 2-Approximation for Tree-Cut Decomposition* **

Eun Jung Kim[1], Sang-il Oum[2], Christophe Paul[3], Ignasi Sau[3], and Dimitrios M. Thilikos[3,4,5]

[1] CNRS, LAMSADE, Paris, France.
eunjungkim78@gmail.com
[2] Department of Mathematical Sciences, KAIST, Daejeon, South Korea.
sangil@kaist.edu
[3] CNRS, Université de Montpellier, LIRMM, Montpellier, France.
christophe.paul@lirmm.fr, ignasi.sau@lirmm.fr, sedthilk@thilikos.info
[4] Department of Mathematics, University of Athens, Greece.
[5] Computer Technology Institute Press "Diophantus", Patras, Greece.

Abstract. The tree-cut width of a graph is a graph parameter defined by Wollan *[J. Comb. Theory, Ser. B, 110:47–66, 2015]* with the help of tree-cut decompositions. In certain cases, tree-cut width appears to be more adequate than treewidth as an invariant that, when bounded, can accelerate the resolution of intractable problems. While designing algorithms for problems with bounded tree-cut width, it is important to have a parametrically tractable way to compute the exact value of this parameter or, at least, some constant approximation of it. In this paper we give a parameterized 2-approximation algorithm for the computation of tree-cut width; for an input n-vertex graph G and an integer w, our algorithm either confirms that the tree-cut width of G is more than w or returns a tree-cut decomposition of G certifying that its tree-cut width is at most $2w$, in time $2^{O(w^2 \log w)} \cdot n^2$. Prior to this work, no *constructive* parameterized algorithms, even approximated ones, existed for computing the tree-cut width of a graph. As a consequence of the Graph Minors series by Robertson and Seymour, only the *existence* of a decision algorithm was known.

Keywords: Fixed-Parameter Tractable algorithm; tree-cut width; approximation algorithm.

* The research of the last author was co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF), Research Funding Program: ARISTEIA II. The second author was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2011-0011653).

** An extended abstract of this article appeared in [16]

1 Introduction

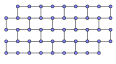
One of the most popular ways to decompose a graph into smaller pieces is given by the notion of a tree decomposition. Intuitively, a graph G has a tree decomposition of small width if it can be decomposed into small (possibly overlapping) pieces that are altogether arranged in a tree-like structure. The *width* of such a decomposition is defined as the minimum size of these pieces. The graph invariant of *treewidth* corresponds to the minimum width of all possible tree decompositions and, that way, serves as a measure of the topological resemblance of a graph to the structure of a tree. The importance of tree decompositions and treewidth in graph algorithms resides in the fact that a wide family of NP-hard graph problems admits FPT-algorithms, i.e., algorithms that run in $f(w) \cdot n^{O(1)}$ steps, when parameterized by the treewidth w of their input graph. According to the celebrated theorem of Courcelle, for every problem that can be expressed in Monadic Second Order Logic (MSOL) [5] it is possible to design an $f(w) \cdot n$-step algorithm on graphs of treewidth at most w. Moreover, towards improving the parametric dependence, i.e., the function f, of this algorithm for specific problems, it is possible to design tailor-made dynamic programming algorithms on the corresponding tree decompositions. Treewidth has also been important from the combinatorial point of view. This is mostly due to the celebrated "*planar graph exclusion theorem*" [18, 19]. This theorem asserts that:

> (*) *Every graph that does not contain some fixed wall*[1] *as a topological minor*[2] *has bounded treewidth.*

The above result had a considerable algorithmic impact as every problem for which a negative (or positive) answer can be certified by the existence of some sufficiently big wall in its input, is reduced to its resolution on graphs of bounded treewidth [6, 9, 10]. This induced a lot of research on the derivation of fast parameterized algorithms that can construct (optimally or approximately) these decompositions. For instance, according to [1], treewidth can be computed in $2^{O(OPT^3)} \cdot n$ steps while, more recently, a 5-approximation for treewidth was given in [2] that runs in $2^{O(OPT)} \cdot n$ steps.

Unfortunately, the aforementioned success stories about treewidth have some natural limitations. In fact, it is not always possible to use treewidth for improving the tractability of NP-hard problems. In particular, there are interesting cases of problems where no such an FPT-algorithm is expected to exist [7, 8, 13]. Therefore, it is a natural question whether there are alternative, but still general, graph invariants that can provide tractable parameterizations for such problems.

A promising candidate in this direction is the graph invariant of *tree-cut width* that was recently introduced by Wollan in [27]. Tree-cut width can be

[1] We avoid the formal definition of a wall here. Instead, we provide the following image  that, we believe, provides the necessary intuition.

[2] A graph H is a *topological minor* of a graph G if a subdivision of H is a subgraph of G.

seen as an "edge" analogue of treewidth. It is defined using a different type of decompositions, namely, tree-cut decompositions that are roughly tree-like partitions of a graph into mutually disjoint pieces such that both the size of some "essential" extension of these pieces and the number of edges crossing two neighboring pieces are bounded (see Section 2 for the formal definition). Our first result is that it is NP-hard to decide, given a graph G and an integer w, whether the input graph G has tree-cut width at most w. This follows from a reduction from the Min Bisection problem that is presented in Subsection 2.2. This encourages us to consider a parameterized algorithm for this problem.

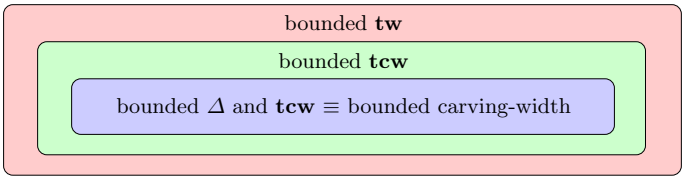


Fig. 1. The relations between classes with bounded treewidth (**tw**) and tree-cut width (**tcw**).

Another tree-like parameter that can be seen as an edge-counterpart of treewidth is *carving-width*, defined in [22]. It is known that a graph has bounded carving-width if and only if both its treewidth and its maximum degree are bounded. We stress that this is not the case for tree-cut width, which can also capture graphs with unbounded maximum degree and, thus, is more general than carving-width. There are two reasons why tree-cut width might be a good alternative for treewidth. We expose them below.

(1) Tree-cut width as a parameter. From now on we denote by $\mathbf{tcw}(G)$ (resp. $\mathbf{tw}(G)$) the tree-cut width (resp. treewidth) of a graph G. As it is shown in [27] $\mathbf{tcw}(G) = O(\mathbf{tw}(G) \cdot \Delta(G))$. Moreover, in [11], it was proven that $\mathbf{tw}(G) = O((\mathbf{tcw}(G))^2)$ and in Subsection 2.3, we prove that the latter upper bound is asymptotically tight. The graph class inclusions generated by the aforementioned relations are depicted in Fig. 1. As tree-cut width is a "larger" parameter than treewidth, one may expect that some problems that are intractable when parameterized by treewidth (known to be W[1]-hard or open) become tractable when parameterized by tree-cut width. Indeed, some recent progress on the development of a dynamic programming framework for tree-cut width (see [11]) confirms that assumption. According to [11], such problems include Capacitated Dominating Set problem, Capacitated Vertex Cover [7], and Balanced Vertex-Ordering problem. We expect that more problems will fall into this category.

(2) Combinatorics of tree-cut width. In [27] Wollan proved the following counterpart of (*):

(**) *Every graph that does not contain some fixed wall as an immersion*[3] *has bounded tree-cut width.*

Notice that (*) yields (**) if we replace "topological minor" by "immersion" and "treewidth" by "tree-cut width". This implies that tree-cut width has combinatorial properties analogous to those of treewidth. It follows that every problem where a negative (or positive) answer can be certified by the existence of a wall as an immersion, can be reduced to the design of a suitable dynamic programming algorithm for this problem on graphs of bounded tree-cut width.

Computing tree-cut width. It follows that designing dynamic programming algorithms on tree-cut decompositions might be a promising task when this is not possible (or promising) on tree decompositions. Clearly, this makes it imperative to have an efficient algorithm that, given a graph G and an integer w, constructs tree-cut decompositions of width at most w or reports that this is not possible. Interestingly, an $f(w) \cdot n^3$-time algorithm for the *decision version* of the problem is known to *exist* but this is not done in a constructive way. Indeed, for every fixed w, the class of graphs with tree-cut width at most w is closed under immersions [27]. By the fact that graphs are well-quasi-ordered under immersions [20], for every w, there exists a *finite* set $\mathcal{R}_w$ of graphs such that G has tree-cut width at most w if and only if it does not contain any of the graphs in $\mathcal{R}_w$ as an immersion. From [14], checking whether an h-vertex graph H is contained as an immersion in some n-vertex graph G can be done in $f(w) \cdot n^3$ steps. It follows that there *exists* an FPT-algorithm checking whether the tree-cut width of a graph is at most w, where w is the parameter.

In fact, an $f(w) \cdot n$-time algorithm for tree-cut width can be obtained in the following way: it is known that if the tree-cut width is at most w, then the treewidth is at most $2w^2 + 3w$ (see Proposition 1). Then, using the algorithm of [2], one can build a tree decomposition of width at most $O(w^2)$ in time $g(w) \cdot n$, or correctly decide that the tree-cut width is larger than w (see Proposition 2). Since deciding whether G contains a fixed graph H as an immersion can be expressed as an Monadic Second Order Logic formula, in time $(f(w) \cdot n)$ one can verify whether G contains any graph in the set $\mathcal{R}_w$ as an immersion by the result of [5].

The *construction* of the aforementioned algorithms requires the explicit knowledge of the set $\mathcal{R}_w$ for every w, which is not provided by the results in [20]. Even if we knew $\mathcal{R}_w$, it is not clear how to construct a tree-cut decomposition of width at most w, if one exists.

In this paper we make a first step towards a constructive FPT-algorithm for tree-cut width by giving an FPT 2-approximation for it. Given a graph G and an integer w, our algorithm either reports that G has tree-cut width more than w or outputs a tree-cut decomposition of width at most $2w$ in $2^{O(w^2 \log w)} n^2$ steps. The algorithm is presented in Section 3.

[3] A graph H is an *immersion* of a graph G if H can be obtained from some subgraph of G after replacing edge-disjoint paths with edges.

2 Problem definition and preliminary results

Unless specified otherwise, every graph in this paper is undirected and loopless and may have multiple edges. By $V(G)$ and $E(G)$ we denote the vertex set and the edge set, respectively, of a graph G. Given a vertex $x \in V(G)$, the *neighborhood* of x is $N(x) = \{y \in V(G) \mid xy \in E(G)\}$. Given two disjoint sets X and Y of $V(G)$, we denote $\delta_G(X, Y) = \{xy \in E(G) \mid x \in X, y \in Y\}$. For a subset X of $V(G)$, we define $\partial_G(X) = \{x \in X \mid N(x) \setminus X \neq \emptyset\}$. We may drop the lower index G when it is clear from the context.

The set of all (positive, respectively) natural numbers is denoted as $\mathbb{N}$ ($\mathbb{N}^+$ respectively).

2.1 Tree-cut width and treewidth

Tree-cut width. A *tree-cut decomposition* of G is a pair $(T, \mathcal{X})$ where T is a tree and $\mathcal{X} = \{X_t \subseteq V(G) \mid t \in V(T)\}$ such that

- $X_t \cap X_{t'} = \emptyset$ for all distinct t and t' in $V(T)$,
- $\bigcup_{t \in V(T)} X_t = V(G)$.

From now on we refer to the vertices of T as *nodes*. The sets in $\mathcal{X}$ are called the *bags* of the tree-cut decomposition. Observe that the conditions above allow to assign an empty bag for some node of T.

Let $L(T)$ be the set of leaf nodes of T. For every tree-edge $e = \{u, v\}$ of $E(T)$, we let T_u^e and T_v^e be the subtrees of $T \setminus e$ which contain u and v, respectively. We drop the upper index e when the relevant edge e is clear from the context.

We define the *adhesion* of a tree-edge $e = \{u, v\}$ of T as follows:

$$\delta^T(e) = \delta_G(\bigcup_{t \in V(T_u)} X_t, \bigcup_{t \in V(T_v)} X_t).$$

Dissolving a vertex x of degree two with exactly two neighbors y and z is the operation of removing x and adding the edge $\{y, z\}$. In case the edge $\{y, z\}$ already exists, its multiplicity is increased by one. For a graph G and a set $X \subseteq V(G)$, the *3-center* of (G, X) is the graph obtained from G by repeatedly applying the following operations: for a vertex $v \in V(G) \setminus X$, we dissolve v if it has two neighbors and degree 2, and remove v if it has degree at most 2 and one neighbor. It is shown in [27] that repeatedly applying these operations lead to a unique graph regardless of the order of the operations, thus the 3-center of (G, X) is well-defined.

Given a tree-cut decomposition $(T, \mathcal{X})$ of G and node $t \in V(T)$, let $T_1, \ldots, T_\ell$ be the connected components of $T \setminus t$. The *torso* of G *at* t, denoted by H_t, is the graph obtained from G by identifying each non-empty vertex set $Z_i := \bigcup_{b \in V(T_i)} X_b$ into a single vertex z_i (in this process, parallel edges are kept). We denote by $\bar{H}_t$ the 3-center of (H_t, X_t). Then the *width* of $(T, \mathcal{X})$ equals

$$\max\left(\{|\delta^T(e)| : e \in E(T)\} \cup \{|V(\bar{H}_t)| : t \in V(T)\}\right).$$

The *tree-cut width* of G, or $\mathbf{tcw}(G)$ in short, is the minimum width of $(T, \mathcal{X})$ over all tree-cut decompositions $(T, \mathcal{X})$ of G.

The following definitions will be used in the approximation algorithm. Let $(T, \mathcal{X})$ be a tree-cut decomposition of G. It is *non-trivial* if it contains at least two non-empty bags, and *trivial* otherwise. We will assume that every leaf of a tree-cut decomposition has a non-empty bag. The *internal-width* of a non-trivial tree-cut decomposition $(T, \mathcal{X})$ is

$$\mathbf{in\text{-}tcw}(T, \mathcal{X}) = \max\left(\{|\delta^T(e)| : e \in E(T)\} \cup \{|V(\bar{H}_t)| : t \in V(T) \setminus L(T)\}\right).$$

If $(T, \mathcal{X})$ is trivial, then we set $\mathbf{in\text{-}tcw}(T, \mathcal{X}) = 0$.

Our decision problem corresponding to tree-cut width is the following:

Tree-cut Width
Input: a graph G and a non-negative integer k.
Question: $\mathbf{tcw}(G) \leqslant k$?

Treewidth. A *tree decomposition* of a graph G is a pair $(T, \mathcal{Y})$, where T is a tree and $\mathcal{Y} = \{Y_x : x \in V(T)\})$ is a collection of subsets of $V(G)$, such that

- $\bigcup_{x \in V(T)} Y_x = V(G)$;
- for every edge $\{u, v\} \in E(G)$ there exists $x \in V(T)$ such that $u, v \in Y_x$; and
- for every vertex $u \in V(G)$ the set of nodes $\{x \in V(T) : u \in Y_x\}$ induces a subtree of T.

The vertices of T are called *nodes* of $(T, \mathcal{Y})$ and the sets Y_x are called bags. The *width* of a tree decomposition is the size of the largest bag minus one. The *treewidth* of a graph, denoted by $\mathbf{tw}(G)$, is the smallest width of a tree decomposition of G.

2.2 Computing tree-cut width is NP-complete

We prove that Tree-cut Width is NP-hard by a polynomial-time reduction from Min Bisection, which is known to be NP-hard [12]. The input of Min Bisection is a graph G and a non-negative integer k, and the question is whether there exists a bipartition (V_1, V_2) of $V(G)$ such that $|V_1| = |V_2|$ and $|\delta_G(V_1, V_2)| \leqslant k$.

Theorem 1. Tree-cut Width *is* NP*-complete.*

Proof: It is easy to see that Tree-cut Width is in NP. We present a reduction from Min Bisection to Tree-cut Width (see Fig. 2). Let (G, k) be an instance of Min Bisection, where G is a simple graph on n vertices and n is even. We may assume that $k \leqslant n^2$ since otherwise, the instance is trivially Yes. Also we assume that $n \geqslant 2$. We create an instance (G', w) with $w = \frac{n^3}{2} + k$ as follows. The vertex set $V(G')$ consists of a set $V(G)$ of size n, a set Q of size $w - 2$, and the set $C_{x,y}$ of size $w + 1$ for every pair $x, y \in Q$. Edges are added so that:

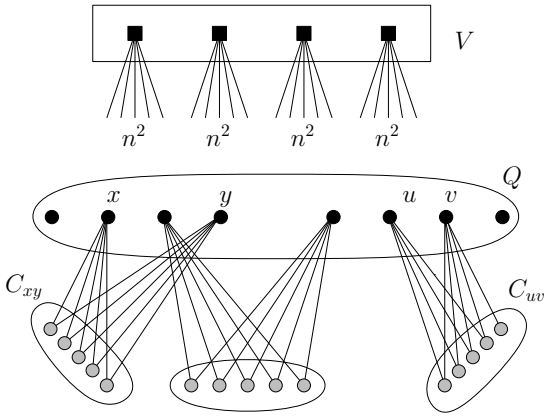


Fig. 2. The graph G' in the transformation of the instances of Min Bisection to equivalent instances of Tree-cut Width.

- $G'[V] = G$.
- For every pair $x, y \in Q$, all vertices of $C_{x,y}$ are adjacent with both x and y.
- Each $x \in V$ is adjacent with n^2 (arbitrarily chosen) vertices of Q.

We now proceed with the proof of the correctness of the above reduction. Suppose that (G, k) is a Yes-instance to Min Bisection with a bipartition (V_1, V_2). We construct a tree-cut decomposition $(T, \mathcal{X})$ in which $V(T)$ contains three nodes t_1, t_2, q and some additional nodes as follows: the tree T forms a star with q as the center and all other nodes as leaves. We have $X_{t_i} = V_i$ for $i = 1, 2$, $X_q = Q$ and each vertex of $\bigcup_{x,y \in Q} C_{x,y}$ forms a singleton bag. It is not difficult to verify that $(T, \mathcal{X})$ is a tree-cut decomposition of G' whose width is w. In particular, notice that $|V(\bar{H}_q)| = |Q| + 2 = w$ and $|\delta^T(\{t_i, q\})| = \frac{n}{2} \cdot n^2 + k = w$ for $i \in \{1, 2\}$.

Conversely, suppose that G' admits a tree-cut decomposition $(T, \mathcal{X})$ of width at most w. Any two vertices $x, y \in Q$ must be in the same bag since they are connected by $w + 1$ disjoint paths via $C_{x,y}$. Hence, there exists a tree node, say q, in T such that $Q \subseteq X_q$.

Consider the set $\mathcal{C} = \{T_1, \ldots, T_\ell\}$ of the connected components of $T \setminus \{q\}$ and let e_i be the tree-edge between T_i and q. Since $w \geqslant |V(\bar{H}_q)| \geqslant |Q| = w - 2$, there are at most two tree-edges among $e_1, \ldots, e_\ell$ such that $|\delta^T(e_i)| \geqslant 3$: indeed, for any tree-edge e_i with $|\delta^T(e_i)| \geqslant 3$, the vertex corresponding to $\bigcup_{b \in V(T_i)} X_b$ in the torso of G at q is not eliminated when creating the 3-center $\bar{H}_q$ of (H_q, X_q). This implies that $\bar{H}_q$ has at least $w+1$ vertices, a contradiction. This means that there are at most two subtrees among $T_1, \ldots, T_\ell$ such that $V \cap \bigcup_{t \in V(T_i)} X_t \neq \emptyset$. From the fact that $|X_q| \leqslant |V(\bar{H}_q)| \leqslant w$, $Q \subseteq X_q$ and $|Q| = w - 2$, we have $|X_q \setminus Q| \leqslant 2$ and especially, at most two vertices of V can be contained in X_q. Therefore, there exists at least one subtree T_i such that $V \cap \bigcup_{t \in V(T_i)} X_t \neq \emptyset$. If there is i such that $|V \cap \bigcup_{t \in V(T_i)} X_t| \geqslant \frac{n}{2} + 1$, then $|\delta^T(e_i)| \geqslant (\frac{n}{2} + 1) \cdot n^2 > w$, a contradiction. Hence, we conclude that there are exactly two subtrees, say T_1

and T_2, in $\mathcal{C}$ such that $V \cap \bigcup_{t \in V(T_i)} X_t \neq \emptyset$ for $i \in \{1,2\}$ and for $3 \leqslant i \leqslant \ell$, we have $V \cap \bigcup_{t \in V(T_i)} X_t = \emptyset$. If any vertex of V is contained in X_q, then $\bar{H}_q$ contains $|X_q| + 2 \geqslant w + 1$ vertices, a contradiction. Therefore, the sets $V \cap \bigcup_{t \in V(T_1)} X_t$ and $V \cap \bigcup_{t \in V(T_2)} X_t$ make a bipartition of V. Observe that the two sets are of equal size due to $|V \cap \bigcup_{t \in V(T_i)} X_t| \leqslant \frac{n}{2}$. Let us call this bipartition $\{V_1, V_2\}$. Observe that $\delta^T(e_i) \supseteq \delta_G(V_i, Q) \cup \delta_G(V_1, V_2)$, thus $\delta^T(e_i)$ contains at least $\frac{n}{2} \cdot n^2 + |\delta_G(V_1, V_2)|$ edges for $i = 1, 2$. As $|\delta^T(e_1)| \leqslant w$, it follows $|\delta_G(V_1, V_2)| \leqslant k$. Therefore, (G, k) is YES-instance to MIN BISECTION which completes the proof. □

2.3 Tree-cut width vs treewidth

In this section we investigate the relation between treewidth and tree-cut width. The following was proved in [11].

Proposition 1. *For a graph of tree-cut width at most w, its treewidth is at most $2w^2 + 3w$.*

In the rest of this subsection we prove that the bound of Proposition 1 is asymptotically optimal. For this we need some definitions.

Let G be a graph. Two subgraphs X and Y of G *touch* each other if either $V(X) \cap V(Y) \neq \emptyset$ or there is an edge $e = \{x, y\} \in E(G)$ with $x \in V(X)$ and $y \in V(Y)$. A *bramble* $\mathcal{B}$ is a collection of connected subgraphs of G pairwise touching each other. The *order* of a bramble $\mathcal{B}$ is the minimum size of a hitting set S of $\mathcal{B}$, that is a set $S \subseteq V(G)$ such that for every $B \in \mathcal{B}$, $S \cap V(B) \neq \emptyset$. In [21], it is shown that the treewidth of a graph equals the maximum order over all brambles of G minus one. Therefore, a bramble of order k is a certificate that the treewidth is at least $k - 1$.

We next define the family of graphs $\mathcal{H} = \{H_w : w \in \mathbb{N}_{\geqslant 1}\}$ as follows. The vertex set of H_w is a disjoint union of w cliques, $Q_1, \ldots, Q_w$, each containing w vertices. For each $1 \leqslant i \leqslant w$, the vertices of Q_i are labeled as (i, j), $1 \leqslant j \leqslant w$. Besides the edges lying inside the cliques Q_i's, we add an edge between $(i, j) \in Q_i$ and $(j, i) \in Q_j$ for every $1 \leqslant i < j \leqslant w$. Notice that the vertex (i, i) does not have a neighbor outside Q_i. The graph H_4 is depicted in Fig. 3.

Lemma 1. *The tree-cut width of H_w is at most $w + 1$.*

PROOF: Consider the tree-cut decomposition $(T, \mathcal{X})$, in which T is a star with t as the center and $q_1, \ldots, q_w$ as leaves. For the bags, we set $X_t = \emptyset$, and $X_{q_i} = Q_i$ for $1 \leqslant i \leqslant w$. It is straightforward to verify that the tree-cut width of $(T, \mathcal{X})$ is $w + 1$. □

Lemma 2. *For any positive integer w, the treewidth of $H_w \in \mathcal{H}$ is at least $\frac{1}{16}w^2 - 1$.*

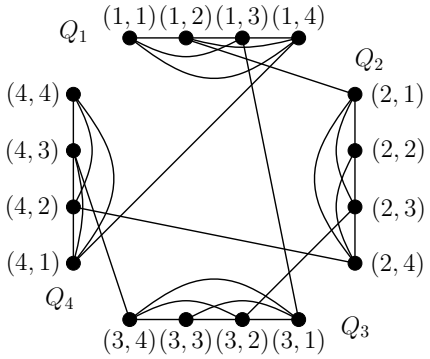


Fig. 3. The graph H_4.

Proof: For notational convenience, we assume that w is even. The argument can be easily extended to the case when w is odd. For $i \in [w]$ and a set $Z \subseteq [w]$, let $B(i, Z)$ denote the set $\{(i, j), (j, i) : j \in Z\}$. We define $\mathcal{B}_w$ as

$$\mathcal{B}_w = \{G[B(i, Z)] : i \in [w], Z \subseteq [w] \setminus \{i\} \text{ and } |Z| = w/2\}.$$

It is easy to verify that each subgraph of $\mathcal{B}_w$ is connected. For any $i \in [w]$ and $Z \subseteq [w] \setminus \{i\}$ such that $|Z| = \frac{1}{2}w$, the number of cliques Q_i, $1 \leqslant i \leqslant w$, with which $B(i, Z)$ has non-empty intersection is at least $\frac{1}{2}w+1$. This means any two elements of $\mathcal{B}_w$ touch each other, and thus $\mathcal{B}_w$ is indeed a bramble. Henceforth, we show that the order of $\mathcal{B}_w$ is at least $\frac{1}{16}w^2$.

Suppose that there is a hitting set S of $\mathcal{B}_w$ with $|S| < \frac{1}{16}w^2$. We define

$$F_S = \{i \in [w] \mid |\{j \in [w] : (j, i) \in V(G) \setminus S\}| \geqslant \frac{3}{4}w\}.$$

Claim 1. $|F_S| > \frac{3}{4}w$.

Proof of the Claim: Suppose that the contrary holds. We use a counting argument to derive a contradiction. The set $V(G) \setminus S$ is partitioned into two sets: $\{(j, i) : j \in [w], i \in F_S\}$ and $\{(j, i) : j \in [w], i \in [w] \setminus F_S\}$. We have

$$|V(G) \setminus S| \leqslant w \cdot |F_S| + \frac{3}{4}w \cdot (w - |F_S|) \leqslant \frac{3}{4}w^2 + \frac{3}{16}w^2 = \frac{15}{16}w^2,$$

contradicting to the assumption that $|S| < \frac{1}{16}w^2$. ◇

Claim 2. *There exists $i^* \in F_S$ such that $|\{j \in [w] : (i^*, j) \in S\}| < \frac{1}{4}w$.*

Proof of the Claim: Suppose the contrary, i.e. we have $|\{j \in [w] : (i, j) \in V(G) \setminus S\}| \leqslant \frac{3}{4}w$ for every $i \in F_S$. Notice that the set $V(G) \setminus S$ is partitioned into $\{(i, j) : i \in F_S, j \in [w]\}$ and $\{(i, j) : i \in [w] \setminus F_S, j \in [w]\}$. Then,

$$|V(G) \setminus S| \leqslant |F_S| \cdot \frac{3}{4}w + (w - |F_S|) \cdot w \leqslant w^2 - \frac{1}{4}w \cdot |F_S| < \frac{13}{16}w^2,$$

where the last inequality follows from Claim 1. This contradicts the assumption that $|S| < \frac{1}{16}w^2$. $\diamond$

Consider some $i^* \in F_S$ satisfying the condition of Claim 2. We observe that the set

$$Z = \{j \in [w] : (j, i^*) \in V(G) \setminus S\} \setminus (\{i^*\} \cup \{j \in [w] : (i^*, j) \in S\})$$

contains at least $\frac{1}{2}w$ vertices by the definition of F_S and Claim 2. Pick any subset Z^* of Z of size exactly $\frac{1}{2}w$. To reach a contradiction, it suffices to show that $B(i^*, Z^*) \cap S = \emptyset$. Indeed, from the fact that $Z^* \subseteq \{j \in [w] : (j, i^*) \in V(G) \setminus S\}$, it follows that

$$\forall j \in Z^* \ (j, i^*) \in V(G) \setminus S. \tag{1}$$

By the definition of Z it follows that $Z^* \cap \{j \in [w] : (i^*, j) \in S\} = \emptyset$, which, implies that

$$\forall j \in Z^* \ (i^*, j) \in V(G) \setminus S. \tag{2}$$

By (1) and (2), we conclude that $B(i^*, Z^*) \cap S = \emptyset$. This completes the proof. $\square$

From Lemmas 1 and 2, we conclude to the following.

Theorem 2. *For every $w \in \mathbb{N}_{\geqslant 1}$ there exists a graph H_w such that $\mathbf{tw}(H_w) = \Omega((\mathbf{tcw}(H_w))^2)$.*

3 The 2-approximation algorithm

We present a 2-approximation of Tree-cut Width running in time $2^{O(w^2 \log w)} \cdot n^2$. As stated in Lemma 3 below, we first observe that computing the tree-cut width of G reduces to computing the tree-cut width of 3-edge-connected graphs. This property can be easily derived from [27, Lemmas 10–11].

Lemma 3. *Given a connected graph G, let $\{V_1, V_2\}$ be a partition of $V(G)$ such that $\delta_G(V_1, V_2)$ is a minimal cut of size at most two and let $w \geqslant 2$ be a positive integer. For $i = 1, 2$, let G_i be the graph obtained from G by identifying the vertex set V_{3-i} into a single vertex v_{3-i}. Then G has tree-cut width at most w if and only if both G_1 and G_2 have tree-cut width at most w.*

Proof: Recall that $\mathbf{tcw}(H) \leqslant \mathbf{tcw}(G)$ if G admits an immersion of H by [27, Lemma 11]. Hence, in order to prove the forward implication, it suffices to prove that G_i is an immersion of G, for $i \in \{1, 2\}$. If $|\delta(V_1, V_2)| = 1$, we can delete all vertices of V_{3-i} except for the single vertex in $N(V_i)$ and obtain G_i. If $|\delta(V_1, V_2)| = 2$, let P be an arbitrary walk between vertices (possibly the same) of $N(V_{3-i})$ in $G[V_{3-i} \cup N(V_{3-i})]$ such that all internal vertices of P are in V_{3-i}. Such P exists since $G[V_{3-i}]$ is connected due to the minimality assumption on $\delta(V_1, V_2)$. By lifting a sequence of edge pairs that share a common end vertices in

V_{3-i} along P until the obtained walk has exactly one vertex of V_{3-i}, and deleting all the vertices and edge of $G[V_{3-i}]$ that are not contained in the obtained walk, we can obtain G_i. Here, *lifting* a pair of edges $e_1 = u_1v$ and $e_2 = u_2v$ with a common end vertex v and $u_1 \neq u_2$ is an operation of removing e_1 and e_2 and adding a new edge u_1u_2.

Conversely, let $(T^i, \mathcal{X}^i)$ be a tree-cut decomposition of G_i of width at most w for $i = 1, 2$, and consider the tree-cut decomposition $(T, \mathcal{X})$ such that $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$ and T is obtained by the disjoint union of T^1 and T^2 after adding an edge between $t_1 \in V(T^1)$ and $t_2 \in V(T^2)$, where t_i is the tree node of T_i containing v_{3-i}, i.e. the vertex obtained by contracting V_{3-i}. We remove v_1 and v_2 from the bags of T.

We claim that $(T, \mathcal{X})$ is a tree-cut decomposition of width at most w. Note first that the adhesion of $(T, \mathcal{X})$ is at most w since $|\delta^T(\{t_1, t_2\})| \leqslant 2$ and the adhesion of $(T^i, \mathcal{X}^i)$ is at most w for $i = 1, 2$. From $|\delta^T(\{t_1, t_2\})| \leqslant 2$, it follows that for $i = 1, 2$, the 3-center of (H_{t_i}, X_{t_i}) of the tree decomposition $(T, \mathcal{X})$ is the same as the 3-center of (H_{t_i}, X_{t_i}) of the tree decomposition $(T^i, \mathcal{X}^i)$. Therefore the width of $(T, \mathcal{X})$ is at most w. □

The proof of the next lemma is easy and is omitted.

Lemma 4. *Let G be a graph and let v be a vertex of G with degree 1 (resp. 2). Let also G' be the graph obtained from G after removing (resp. dissolving) v. Then* $\mathbf{tcw}(G) = \mathbf{tcw}(G')$.

From now on, based on Lemmas 3 and 4, we assume that the input graph is 3-edge-connected. In this special case, the following observation is not difficult to verify. It allows us to work with a slightly simplified definition of the 3-centers in a tree-cut decomposition.

Observation 1. *Let G be a 3-edge-connected graph and let $(T, \mathcal{X})$ be a tree-cut decomposition of G. Consider an arbitrary node t of $V(T)$ and let $\mathcal{T}$ be the set of all connected components T' of $T \setminus t$ such that $\bigcup_{s \in V(T')} X_s \neq \emptyset$. Then $|V(\bar{H}_t)| = |X_t| + |\mathcal{T}|$, that is $|V(\bar{H}_t)| = |V(H_t)|$.*

Proof: Since G is 3-edge-connected, for every node t of $V(T)$ and every connected component T' of $T \setminus t$ such that $\bigcup_{s \in V(T')} X_s \neq \emptyset$, the degree in the torso H_t of the vertex z' resulting from the identification of $\bigcup_{s \in V(T')} X_s$ is at least 3. Therefore, when constructing the 3-center $\bar{H}_t$ of (H_t, X_t), vertex z' does not get dissolved nor removed, yielding that $|V(\bar{H}_t)| = |V(H_t)|$. □

We observe that the proof of Lemma 3 provides a way to construct a desired tree-cut decomposition for G from decompositions of smaller graphs. Given an input graph G for Tree-cut Width, we find a minimal cut (V_1, V_2) with $|\delta(V_1, V_2)| \leqslant 2$ and create a graph G_i as in Lemma 3, with the vertex v_{3-i} marked as distinguished. We recursively find such a minimal cut in the smaller graphs created until either one becomes 3-edge-connected or has at most w vertices.

Therefore, a key feature of an algorithm for Tree-cut Width lies in how to handle 3-edge-connected graphs. Our algorithm iteratively refines a tree-cut decomposition $(T, \mathcal{X})$ of the input graph G and either guarantees that the following invariant is satisfied or returns that $\mathbf{tcw}(G) > \omega$.

Invariant: $(T, \mathcal{X})$ *is a tree-cut decomposition of* G *where* $\mathbf{in\text{-}tcw}(T, \mathcal{X}) \leqslant 2 \cdot w$.

Clearly the trivial tree-cut decomposition satisfies the *Invariant*. A leaf t of T such that $|X_t| \geqslant 2 \cdot \omega$ is called a *large leaf*. At each step, the algorithm picks a large leaf and refines the current tree-cut decomposition by breaking this leaf bag into smaller pieces. The process repeats until we finally obtain a tree-cut decomposition of width at most $2w$, or encounter a certificate that $\mathbf{tcw}(G) > w$.

3.1 Refining a large leaf of a tree-cut decomposition

A large leaf will be further decomposed into a star. To that aim, we will solve the following problem:

Constrained Star-Cut Decomposition
Input: An undirected graph G, an integer $w \in \mathbb{N}$, and a weight function $\gamma : V(G) \to \mathbb{N}$.
Parameter: w.
Question: Determine whether G admits a non-trivial tree-cut decomposition $(T, \mathcal{X})$ such that

1. T is a star with central node t_c and with ℓ leaves for some $\ell \in \mathbb{N}^+$,
2. $\mathbf{in\text{-}tcw}(T, \mathcal{X}) \leqslant w$, and
3. $|X_{t_c}| + \ell \leqslant w$ and for every leaf node t, $\gamma(X_t) := \sum_{v \in X_t} \gamma(v) \leqslant w$,

and output such a tree-cut decomposition if one exists.

We notice that as the output $(T, \mathcal{X})$ of the algorithm is a non-trivial tree-cut decomposition if the instance is Yes, T contains at least two nodes with non-empty bags. Without loss of generality, we intend to find a tree-cut decomposition in which every leaf node is non-empty.

Given a subset $S \subseteq V(G)$, we define the instance of the Constrained Star-Cut Decomposition problem $I(S, G) = (G[S], w, \gamma_S)$ where for every $x \in S$, $\gamma_S(x) = |\delta_G(\{x\}, V(G) \setminus S)|$. Notice that $\gamma_S(x) > 0$ only when $x \in \partial_G(S)$.

Lemma 5. *Let G be a 3-edge-connected graph, $w \geqslant 2$ be an integer, and let $S \subseteq V(G)$ be a set of vertices such that $|S| \geqslant w + 1$ and $|\delta_G(S, V(G) \setminus S)| \leqslant 2w$. If $\mathbf{tcw}(G) \leqslant w$, then $I(S, G) = (G[S], w, \gamma_S)$ is a* Yes*-instance of* Constrained Star-Cut Decomposition.

Proof: Let $(T, \mathcal{X})$ be a non-trivial tree-cut decomposition of G of width at most w. We extend the weight function γ_S on S into γ'_S on $V(G)$ by setting $\gamma'_S(v) = \gamma_S(v)$ for every $v \in S$ and $\gamma'_S(v) = 0$ otherwise. Also, given a subtree T' of T, we let $\gamma'_S(T') = \sum_{t \in V(T')} \sum_{v \in X_t} \gamma'_S(v)$. The idea is to identify a node t_c

of T that will serve as the central node of the star decomposition. The leaves of the star decomposition will result from the contraction of the subtrees of $T \setminus t_c$ containing bags that intersect the set S. To find the node t_c, we orient the edges of T using the following two rules. Given an edge $e = \{x, y\} \in E(T)$:

Rule 1: orient e from x to y if $\gamma'_S(T_y) > w$.
Rule 2: orient e from x to y if $S \cap \bigcup_{t \in V(T_x)} X_t = \emptyset$.

Let $\boldsymbol{T}$ be the resulting orientation of T. Observe that Rule 1 and 2 may leave some edges of T non-oriented.

Claim 3. *For every edge $e = \{x, y\}$ of T, e is oriented either in a single direction or not oriented in $\boldsymbol{T}$.*

Proof of the Claim: Observe that if **Rule 1** orients e from x to y, neither **Rule 1** nor **Rule 2** may orient e in the opposite direction. The former is an immediate consequence of the fact $\gamma'_S(T_x) + \gamma'_S(T_y) = |\delta_G(S, V(G) \setminus S)| \leqslant 2w$. **Rule 2** does not orient e from y to x either: if **Rule 2** does so, we have $S \cap \bigcup_{t \in V(T_y)} X_t = \emptyset$ and since the value $\gamma'_S(v)$ is non-zero only when $v \in S$, we conclude that $\gamma'_S(T_y) = 0$, a contradiction to the assumption that **Rule 1** oriented e from x to y. Moreover, the edge e cannot be oriented in both directions by **Rule 2** since S is non-empty and thus at least one of the sets $\bigcup_{t \in V(T_x)} X_t$ and $\bigcup_{t \in V(T_y)} X_t$ intersects with S. ◇

By Claim 3, $\boldsymbol{T}$ contains at least one node, say t_c, which is not incident to an out-going edge in $\boldsymbol{T}$. Let $T_1, \ldots, T_\ell$ be the connected components of $T \setminus t_c$ containing a node t such that $X_t \cap S \neq \emptyset$. Observe that as $|S| \geqslant w + 1$ and $\mathbf{tcw}(T, \mathcal{X}) \leqslant w$, S cannot be included in a single bag of $(T, \mathcal{X})$ and thereby $\ell \geqslant 1$. Consider the following tree-cut decomposition $(T^*, \mathcal{X}^*)$ of $G[S]$:

- T^* is a star with central node t_c and leaf nodes $t_1, \ldots t_\ell$,
- the bag of node t_c is $X^*_c = X_{t_c} \cap S$,
- for every leaf node $t_i \in V(T^*)$, we set $X^*_i = \bigcup_{t \in V(T_i)} X_t \cap S$.

Observe that $(T^*, \mathcal{X}^*)$ is a tree-cut decomposition of $G[S]$ and since $|S| \geqslant w+1$, it is non-trivial. By construction, as it is obtained from $(T, \mathcal{X})$ by contracting subtrees and removing vertices from bags, we have that $\mathbf{in\text{-}tcw}(T^*, \mathcal{X}^*) \leqslant w$. It remains to prove that $|X_{t_c}| + \ell \leqslant w$ and that $\gamma_S(X_t) \leqslant w$ for every leaf node t. The former inequality directly follows from Observation 1 and the fact that $(T, \mathcal{X})$ is an optimal tree-cut decomposition of G. The latter inequality follows from the fact that t does not have an out-going edge in $\boldsymbol{T}$, in particular, **Rule 1** does not orient any edge incident with t outwardly from t. □

Given a 3-edge-connected graph, applying Lemma 5 on a large leaf of a tree-cut decomposition that satisfies the *Invariant*, we obtain:

Corollary 1. *Let G be a 3-edge-connected graph G such that $\mathbf{tcw}(G) \leqslant w$, and let t be a large leaf of a tree-cut decomposition $(T, \mathcal{X})$ satisfying the Invariant. Then $I(X_t, G) = (G[X_t], w, \gamma_{X_t})$ is a* Yes*-instance of* Constrained Star-Cut Decomposition.

The next lemma shows that if a large leaf bag X_t of a tree-cut decomposition $(T, \mathcal{X})$ satisfying the *Invariant* defines a YES-instance of the CONSTRAINED TREE-CUT DECOMPOSITION problem, then $(T, \mathcal{X})$ can be further refined.

Lemma 6. *Let G be a 3-edge-connected graph G and $(T, \mathcal{X})$ be a tree-cut decomposition of G satisfying the* Invariant. *If $(T^*, \mathcal{X}^*)$ is a solution of* CONSTRAINED STAR-CUT DECOMPOSITION *on the instance $I(X_t, G) = (G[X_t], w, \gamma_{X_t})$ where t is a large leaf of $(T, \mathcal{X})$, then the pair $(\tilde{T}, \tilde{\mathcal{X}})$ where*

- $V(\tilde{T}) = (V(T) \setminus \{t\}) \cup V(T^*)$,
- $E(\tilde{T}) = (E(T) \setminus \{\{t, t'\}\}) \cup E(T^*) \cup \{\{t_c, t'\}\}$, *where t' is the unique neighbor of t in T and t_c is the central node of T^*,*
- $\tilde{\mathcal{X}} = (\mathcal{X} \setminus \{X_t\}) \cup \mathcal{X}^*$

is a tree-cut decomposition of G satisfying the Invariant. *Moreover the number of non-empty bags is strictly larger in $(\tilde{T}, \tilde{\mathcal{X}})$ than in $(T, \mathcal{X})$.*

PROOF: By construction, $(\tilde{T}, \tilde{\mathcal{X}})$ is a tree-cut decomposition of G. The fact that $(T^*, \mathcal{X}^*)$ is non-trivial implies that the number of non-empty bags is strictly larger in $(\tilde{T}, \tilde{\mathcal{X}})$ than in $(T, \mathcal{X})$.

It remains to prove that $\textbf{in-tcw}(\tilde{T}, \tilde{\mathcal{X}}) \leqslant 2 \cdot w$. Since $(T^*, \mathcal{X}^*)$ is a solution to $I(X_t, G)$, we have $|X^*_{t_c}| + \ell \leqslant w$. As G is edge 3-connected, by Observation 1, the torso size at t_c in $(\tilde{T}, \tilde{\mathcal{X}})$ is at most $w+1$, which is at most $2w$. Let us verify that the adhesion of $(\tilde{T}, \tilde{\mathcal{X}})$ is at most $2w$. For this, it suffices to bound the value $|\delta^{\tilde{T}}(e)|$ for the newly created edges $e = \{t_i, t_c\}$, for all $i \in [\ell]$. We have

$$\begin{aligned} |\delta^{\tilde{T}}(\{t_i, t_c\})| &= |\delta_G(\tilde{X}_{t_i}, V(G) \setminus \tilde{X}_{t_i})| \\ &= |\delta_G(\tilde{X}_{t_i}, X_t \setminus \tilde{X}_{t_i})| + |\delta_G(\tilde{X}_{t_i}, V(G) \setminus X_t)| \leqslant 2w. \end{aligned}$$

The inequality follows from the fact that $(T^*, \mathcal{X}^*)$ is a solution to $I(X_t, G)$. More precisely, $|\delta_G(\tilde{X}_{t_i}, X_t \setminus \tilde{X}_{t_i})| \leqslant w$ is implied by the fact that $\textbf{in-tcw}(T^*; \mathcal{X}) \leqslant w$. And $|\delta_G(\tilde{X}_{t_i}, V(G) \setminus X_t)| \leqslant w$ is a consequence of $\gamma_{X_t}(X^*_{t_i}) \leqslant w$ and the construction of γ_{X_t} in $I(X_t, G)$. □

3.2 An FPT algorithm for CONSTRAINED STAR-CUT DECOMPOSITION

Proposition 1 provides a quadratic bound on the treewidth of a graph in terms of its tree-cut width. This allows us to develop a dynamic programming algorithm for solving CONSTRAINED STAR-CUT DECOMPOSITION on graphs of bounded treewidth. To obtain a tree decomposition, we use the 5-approximation FPT-algorithm of the following proposition.

Proposition 2 (see [2]). *There exists an algorithm which, given a graph G and an integer w, either correctly decides that $\textbf{tw}(G) > w$ or outputs a tree decomposition of width at most $5w + 4$ in time $2^{O(w)} \cdot n$.*

If $\mathbf{tcw}(G) \leqslant w$, then by Proposition 1 $\mathbf{tw}(G) \leqslant 2w^2+3w$. Using the algorithm of Proposition 2, we can compute a tree decomposition of G of width at most $5(2w^2 + 3w) + 4$, or correctly conclude that G has tree-cut width larger than w. In what follows, we present an exposition on how to solve Constrained Star-Cut Decomposition in $2^{O(w^2 \cdot \log w)} \cdot n$ steps using a tree decomposition of G of width $O(w^2)$.

A *rooted tree decomposition* $(T, \mathcal{X}, r)$ is a tree decomposition with a distinguished node r selected as the *root*. A *nice tree decomposition* $(T, \mathcal{Y}, r)$ (see [17]) is a rooted tree decomposition, in which every internal node of T has at most two children and the following conditions hold: the bag at the root is $\emptyset$, for each node x with two children y, z (called a *join* node) it holds $Y_x = Y_y = Y_z$, and for each node x with one child y it holds $Y_x = Y_y \cup \{u\}$ (*introduce* node) or $Y_x = Y_y \setminus \{u\}$ (*forget* node) for some $u \in V(G)$. A node of T having no children is called a *leaf* node. We need the following proposition.

Proposition 3 (see [1]). *For any constant $w \geqslant 1$, given a tree decomposition of a graph G of width $\leqslant w$ and $O(|V(G)|)$ nodes, there exists an algorithm that, in $O(|V(G)|)$ time, constructs a nice tree decomposition of G of width $\leqslant w$ and with at most $4|V(G)|$ nodes.*

Lemma 7. *Let (G, w, γ) be an input of* Constrained Star-Cut Decomposition *and let $\mathbf{tw}(G) \leqslant q$. There exists an algorithm that given (G, w, γ) outputs, if one exists, a solution of (G, w, γ) in $2^{O((q+w)\log w)} \cdot n$ steps, or correctly reports that (G, w, γ) is a* No*-instance.*

Proof: From Propositions 2 and 3, we can obtain a nice tree decomposition $(T, \mathcal{Y}, r)$ of G whose width is at most $5q + 4$ in time $2^{O(q)} \cdot n$. We present a dynamic programming algorithm to compute a solution of (G, w, γ) on $(T, \mathcal{Y}, r)$.

For every $1 \leqslant \ell \leqslant w$ and a subset $Z \subseteq V(G)$, we say that a collection $\mathcal{X} = \{X_0, \ldots, X_\ell\}$ of pairwise disjoint subsets (some X_i may be empty) of Z with $\bigcup_{i=0}^{\ell} X_i = Z$ is *ℓ-legitimate* if $|X_0| + \ell \leqslant w$ and for every $i \in \{1, \ldots, \ell\}$, we have $|\delta(X_i, V(G) \setminus X_i)| \leqslant w$ and $\gamma(X_i) \leqslant w$. Observe that (G, w, γ) is a Yes-instance if and only if $V(G)$ admits an ℓ-legitimate collection for some $1 \leqslant \ell \leqslant w$: if $\mathcal{X}$ is an ℓ-legitimate collection for some $1 \leqslant \ell \leqslant w$, then $(T, \mathcal{X})$, where T is a star whose center is labeled by 0 and leaves are bijectively labeled by $\{1, \ldots, \ell\}$, is a solution to the instance (G, w, γ). Conversely, for any solution $(T, \mathcal{X})$ to (G, w, γ), we have $1 \leqslant |V(T)| - 1 \leqslant w$, and the collection of bags $\mathcal{X}$ is ℓ-legitimate with $\ell = |V(T)| - 1$.

Henceforth, we fix ℓ such that $1 \leqslant \ell \leqslant w$. Let Z_t be the vertex set $\bigcup_{t' \in V(T_t)} Y_{t'}$, where T_t is the subtree of T rooted at t. An entry of a table $\mathcal{D}_t$ at node t is a quadruple (ϕ, a, α, β), where $\phi : Y_t \to [0, \ell]$, $a \in \mathbb{N}$, $\alpha : [\ell] \to [0, w]$ and $\beta : [\ell] \to [0, w]$. We intend that for an ℓ-legitimate collection $\mathcal{X} = \{X_0, \ldots, X_\ell\}$ of $V(G)$, a quadruple (ϕ, a, α, β) with the following specification is stored in $\mathcal{D}_t$:

(i) for every $v \in Y_t$, $\phi(v) = i$ if $v \in X_i$,
(ii) $a = |X_0 \cap Z_t|$,
(iii) for every $i \in [\ell]$, $\alpha(i) = \gamma(X_i \cap Z_t)$,

(iv) for every $i \in [\ell]$, $\beta(i) = |\delta(X_i \cap Z_t, Z_t \setminus X_i)|$.

For a property P, the bracket notation $[P]$ takes 1 if the property P holds, and takes 0 otherwise. We present how to create the table $\mathcal{D}_t$ at each node t. In all cases, a computed quadruple (ϕ, a, α, β) shall be discarded if $a+\ell > w$ or, the range of α or β is not included in $[0, w]$. As we present the update procedure at each node type, we prove the following claim: $\mathcal{D}_t$ contains an entry (ϕ, a, α, β) if and only if Z_t admits an ℓ-legitimate collection $\mathcal{X} = \{X_0, X_1, \ldots, X_\ell\}$ such that $v \in X_{\phi(v)}$ for every $v \in Y_t$, $|X_0| = a$, and $\gamma(X_i) = \alpha(i)$, $|\delta(X_i, Z_t \setminus X_i)| = \beta(i)$ for every $i \in [\ell]$. Below, all sums over pairs of vertices are taken over unordered pairs.

Leaf node: For each mapping $\phi : Y_t \to [0, \ell]$, we set

$$\begin{aligned} a &:= \sum_{u \in Y_t} [\phi(u) = 0] \\ \alpha(i) &:= \sum_{u \in Y_t} \gamma(u) \cdot [\phi(u) = i] \qquad \forall i \in [\ell] \\ \beta(i) &:= \sum_{u,v \in Y_t} |\delta(\{u\}, \{v\})| \cdot [\phi(u) = i \wedge \phi(v) \neq i] \qquad \forall i \in [\ell]. \end{aligned}$$

If $\mathcal{D}_t$ contains an entry (ϕ, a, α, β), let X_i be the set of all vertices $v \in Y_t$ with $\phi(v) = i$ for each $i \in [0, \ell]$. Then for $\mathcal{X} = \{X_0, X_1, \ldots, X_\ell\}$, we have $v \in X_{\phi(v)}$ for every $v \in Y_t$, $|X_0| = a$, $\gamma(X_i) = \alpha(i)$ and $|\delta(X_i, Z_t \setminus X_i)| = \beta(i)$. The collection $\mathcal{X}$ is ℓ-legitimate since (ϕ, a, α, β) has not been discarded. Conversely, for an ℓ-legitimate collection $\mathcal{X} = \{X_0, X_1, \ldots, X_\ell\}$, let $\phi : Y_t \to [0, \ell]$ be a mapping such that $\phi(v) = i$ if $v \in X_i$, which is well-defined. Notice that for ϕ, the update procedure at a leaf node ensures that $a = |X_0|$, and $\alpha(i) = \gamma(X_i)$, $\beta(i) = |\delta(X_i, Z_t \setminus X_i)|$ for every $i \in [\ell]$. Since $\mathcal{X}$ is ℓ-legitimate, the entry (ϕ, a, α, β) is in $\mathcal{D}_t$. Therefore, the claim holds when t is a leaf node.

Now we consider a non-leaf node t assuming that the claim holds for all children t' of t.

Forget node: Let $Y_t = Y_{t'} \setminus \{u\}$, where t' is a child of t. For each quadruple $(\phi', a', \alpha', \beta') \in \mathcal{D}_{t'}$, we add the tuple $(\phi, a, \alpha, \beta) = (\phi'|_{Y_t}, a', \alpha', \beta')$ to $\mathcal{D}_t$.

If $(\phi, a, \alpha, \beta) \in \mathcal{D}_t$, then there exists an entry $(\phi', a, \alpha, \beta) \in \mathcal{D}_{t'}$ such that $\phi'|_{Y_t} = \phi$. By induction hypothesis, $Z_{t'} = Z_t$ admits an ℓ-legitimate collection $\mathcal{X}$ such that $v \in X_{\phi(v)}$ for every $v \in Y_t \subseteq Y_{t'}$, $|X_0| = a$, and $\gamma(X_i) = \alpha(i)$, $|\delta(X_i, Z_t \setminus X_i)| = \beta(i)$ for every $i \in [\ell]$. The converse implication can be similarly verified.

Introduce node: Let $Y_t = \{u\} \cup Y_{t'}$, where t' is a child of t. For each quadruple $(\phi', a', \alpha', \beta') \in \mathcal{D}_{t'}$ and for each $j \in [0, \ell]$, we build a quadruple (ϕ, a, α, β) as

follows:

$$\begin{aligned}
\phi &:= \phi' \cup (u, j) \\
a &:= a' + [\phi(u) = 0] \\
\alpha(i) &:= \alpha'(i) + \gamma(u) \cdot [\phi(u) = i] \qquad \forall i \in [\ell] \\
\beta(i) &:= \beta'(i) + \begin{cases} \sum_{v \in Y_{t'}} |\delta(\{u\}, \{v\})| \cdot [\phi(v) \neq \phi(u)] & \text{if } i = \phi(u) \\ \sum_{v \in Y_{t'}} |\delta(\{u\}, \{v\})| \cdot [\phi(v) = i] & \text{otherwise.} \end{cases}
\end{aligned}$$

To see that the claim holds, suppose $(\phi, a, \alpha, \beta) \in \mathcal{D}_t$. Then the update procedure ensures that $(\phi', a', \alpha', \beta') \in \mathcal{D}_{t'}$, where $\phi' = \phi|_{Y_{t'}}$ and a, α', β' satisfy the above equations. By induction hypothesis, $Z_{t'}$ admits an ℓ-legitimate collection $\mathcal{X}' = \{X'_0, X'_1, \ldots, X'_\ell\}$ such that $v \in X'_{\phi'(v)}$ for every $v \in Y_{t'}$, $|X'_0| = a'$, and $\gamma(X'_i) = \alpha'(i)$, $|\delta(X'_i, Z_{t'} \setminus X'_i)| = \beta'(i)$ for every $i \in [\ell]$. Consider the collection $\mathcal{X} = \{X_0, X_1, \ldots, X_\ell\}$, where $X_{\phi(u)} = X'_{\phi(u)} \cup \{u\}$ and $X_i = X'_i$ for every $i \neq \phi(u)$. Clearly, for every $v \in Y_t$, we have $v \in X_{\phi(v)}$ and also $|X_0| = |X'_0| + [\phi(u) = 0] = a' + [\phi(u) = 0] = a$. Observe, for every $i \in [\ell]$

$$\gamma(X_i) = \gamma(X'_i) + \gamma(u) \cdot [\phi(u) = i] = \alpha'(i) + \gamma(u) \cdot [\phi(u) = i] = \alpha(i),$$

and from $Z_t \setminus X_{\phi(u)} = Z_{t'} \setminus X'_{\phi(u)}$ and $\delta(\{u\}, Z_{t'} \setminus X'_{\phi(u)}) = \delta(\{u\}, Y_{t'} \setminus X'_{\phi(u)})$, we have

$$\begin{aligned}
|\delta(X_{\phi(u)}, Z_t \setminus X_{\phi(u)})| &= |\delta(X'_{\phi(u)}, Z_t \setminus X_{\phi(u)})| + |\delta(\{u\}, Z_t \setminus X_{\phi(u)})| \\
&= |\delta(X'_{\phi(u)}, Z_{t'} \setminus X'_{\phi(u)})| + |\delta(\{u\}, Y_{t'} \setminus X'_{\phi(u)})| \\
&= \beta'(\phi(u)) + \sum_{v \in Y_{t'}} |\delta(\{u\}, \{v\})| \cdot [\phi(v) \neq \phi(u)] = \beta(\phi(u)).
\end{aligned}$$

For every $i \in [\ell] \setminus \phi(u)$, we have $X_i = X'_i$ and $\delta(X_i, Z_t \setminus X_i) = \delta(X'_i, Z_{t'} \setminus X'_i) \uplus \delta(X'_i, \{u\})$. It follows that

$$\begin{aligned}
|\delta(X_i, Z_t \setminus X_i)| &= |\delta(X'_i, Z_{t'} \setminus X'_i)| + |\delta(X'_i, \{u\}))| \\
&= \beta'(i) + \sum_{v \in Y_{t'}} |\delta(\{u\}, \{v\})| \cdot [\phi(v) \neq \phi(u)] = \beta(\phi(u)).
\end{aligned}$$

It follows that $\mathcal{X}$ is an ℓ-legitimate collection satisfying the requirement. The converse implication can be similarly verified.

Join node: Let t_1 and t_2 be the two children of t. For each pair of quadruples $(\phi_1, a_1, \alpha_1, \beta_1) \in \mathcal{D}_{t_1}$ and $(\phi_2, a_2, \alpha_2, \beta_2) \in \mathcal{D}_{t_2}$ such that $\phi_1 = \phi_2$, we compute

an entry (ϕ, a, α, β) as follows:

$$\begin{aligned}
\phi &:= \phi_1 (= \phi_2) \\
a &:= a_1 + a_2 - \sum_{v \in Y_t} [\phi(v) = 0] \\
\alpha(i) &:= \alpha_1(i) + \alpha_2(i) - \sum_{u \in Y_t} \gamma(u) \cdot [\phi(u) = i] \qquad \forall i \in [\ell] \\
\beta(i) &:= \beta_1(i) + \beta_2(i) - \sum_{u,v \in Y_t} |\delta(\{u\}, \{v\})| \cdot [\phi(u) = i \wedge \phi(v) \neq i] \qquad \forall i \in [\ell].
\end{aligned}$$

Suppose $(\phi, a, \alpha, \beta) \in \mathcal{D}_t$. Then there are two entries $(\phi_1, a_1, \alpha_1, \beta_1) \in \mathcal{D}_{t_1}$ and $(\phi_2, a_2, \alpha_2, \beta_2) \in \mathcal{D}_{t_2}$ satisfying the above equations. Let $\mathcal{X}' = \{X'_0, X'_1, \ldots, X'_\ell\}$ and $\mathcal{X}'' = \{X''_0, X''_1, \ldots, X''_\ell\}$ be ℓ-legitimate collections of Z_{t_1} and Z_{t_2}, respectively, guaranteed by induction hypothesis. We define $X_i = X'_i \cup X_i$ for every $i \in [0, \ell]$ and $\mathcal{X} = \{X_0, X_1, \ldots, X_\ell\}$. Observe that $\bigcup_{i=0}^{\ell} X_i = Z_{t_1} \cup Z_{t_2} = Z_t$ and for any $i \neq j$, we have $X_i \cap X_j = \emptyset$ since $X_i \cap X_j \subseteq Z_{t_1} \cap Z_{t_2} = Y_t$ and $\phi_1 = \phi_2$. Let us verify that $\mathcal{X}$ is a desired ℓ-legitimate collection.

For every $v \in Y_t$, we have $v \in X'_{\phi_1(v)} \subseteq X_{\phi(v)}$. From $X_0 = X'_0 \cup X''_0$, it follows $|X_0| = |X'_0| + |X''_0| - |X'_0 \cap X''_0| = a_1 + a_2 - \sum_{v \in Y_t} [\phi(v) = 0] = a$. For every $i \in [\ell]$, observe that $\gamma(X_i) = \gamma(X'_i) + \gamma(X''_i) - \gamma(X'_i \cap X''_i) = \alpha_1(i) + \alpha_2(i) - \sum_{v \in Y_t} \gamma(v) \cdot [\phi(v) = i] = \alpha(i)$. From $Z_t = Z_{t_1} \cup Z_{t_2}$, we observe the following for every $i \in [\ell]$,

$$\begin{aligned}
\delta(X_i, Z_t \setminus X_i) &= \delta_{G[Z_{t_1}]}(X_i \cap Z_{t_1}, (Z_t \setminus X_i) \cap Z_{t_1}) \cup \\
&\qquad \delta_{G[Z_{t_2}]}(X_i \cap Z_{t_2}, (Z_t \setminus X_i) \cap Z_{t_2}) \\
&= \delta(X'_i, Z_{t_1} \setminus X'_i) \cup \delta(X''_i, Z_{t_2} \setminus X''_i).
\end{aligned}$$

Notice that in the last equality, the edges between $X'_i \cap X''_i = X_i \cap Y_t$ and $(Z_{t_1} \setminus X'_i) \cap (Z_{t_2} \setminus X''_i)$ contribute twice to the union and other edges contribute exactly once. For the latter set, we have $(Z_{t_1} \setminus X'_i) \cap (Z_{t_2} \setminus X''_i) \subseteq (Z_{t_1} \cap Z_{t_2}) \setminus (X'_i \cup X''_i) = Y_t \setminus X_i$. Furthermore, from $Y_t \subseteq Z_{t_1}$ and $X_i \supseteq X'_i$, we have $Y_t \setminus X_i \subseteq Z_{t_1} \setminus X'_i$. Likewise, $Y_t \setminus X_i \subseteq Z_{t_2} \setminus X''_i$ holds as well. Therefore, $(Z_{t_1} \setminus X'_i) \cap (Z_{t_2} \setminus X''_i) = Y_t \setminus X_i$. It follows that

$$\begin{aligned}
|\delta(X_i, Z_t \setminus X_i)| &= \beta_1(i) + \beta_2(i) - \sum_{u,v \in Y_t} \delta(\{u\}, \{v\}) \cdot [\phi(u) = i, \phi(v) \neq i] \\
&= \beta(i).
\end{aligned}$$

This implies that $\mathcal{X}$ is ℓ-legitimate with the desired property. The opposite direction of the claim can be verified in a similar fashion.

Our claim especially implies that (G, w, γ) is Yes to Constrained Star-Cut Decomposition if and only if $\mathcal{D}_r$ contains an entry (ϕ, a, α, β). By standard backtracking, one can construct a solution to (G, w, γ) if $\mathcal{D}_r$ is non-empty.

It remains to see the running time of the algorithm. There are at most $(\ell + 1)^{|Y_t|} \cdot (w+1) \cdot (w+1)^{2\ell} = (w+1)^{O(q+w)}$ entries in $\mathcal{D}_t$ at node t. Each entry can be

computed in time $|Y_t|^{O(1)}$. The running time for computing $\mathcal{D}_t$ is $O(|\mathcal{D}_t| \cdot |Y_t|^{O(1)})$ when t is a leaf, introduce, or forget node. For a join node t with children t_1 and t_2, the running time is $O(|\mathcal{D}_{t_1}| \cdot |\mathcal{D}_{t_2}| \cdot |Y_t|^{O(1)})$. As we perform dynamic programming for $\ell = 1, \ldots, w$, the entire algorithm runs in time $(w+1)^{O(q+w)} \cdot n = 2^{O((q+w)\log w)} \cdot n$ as stated. □

3.3 Piecing everything together

We now present a 2-approximation algorithm for Tree-cut Width leading to the following result.

Theorem 3. *There exists an algorithm that, given a graph G and $w \in \mathbb{N}$, either outputs a tree-cut decomposition of G with width at most $2w$ or correctly reports that G has tree-cut width larger than w in $2^{O(w^2 \cdot \log w)} \cdot n^2$ steps.*

Proof: Recall that, by Lemmas 3 and 4, we can assume that G is 3-edge-connected. If not, we iteratively decompose G into 3-edge-connected components using the linear-time algorithm of [26]. A tree-cut decomposition of G can be easily built from the tree-cut decomposition of its 3-edge-connected components using Lemma 3. As mentioned earlier, the trivial tree-cut decomposition satisfies the *Invariant*. Let $(T, \mathcal{X})$ be a tree-cut decomposition satisfying the *Invariant*. As long as the current tree-cut decomposition $(T, \mathcal{X})$ contains a large leaf ℓ, the algorithm applies the following steps repeatedly:

1. Let $X_\ell \in \mathcal{X}$ be the bag associated to a large leaf ℓ. Compute a nice tree decomposition of $G[X_\ell]$ of width at most $O(w^2)$ in $2^{O(w^2)} \cdot n$ time. If such a decomposition does not exist, as $G[X_\ell]$ is a subgraph of G, Proposition 1 implies $\mathbf{tcw}(G) > w$ and the algorithm stops.
2. Solve Constrained Star-Cut Decomposition on $I(X_t, G)$ using the dynamic programming of Lemma 7 for $q = O(w^2)$ in time $2^{O(w^2 \cdot \log w)} \cdot n$.
3. If $I(X_t, G)$ is a No-instance, then by Corollary 1, the algorithm correctly reports that $\mathbf{tcw}(G) > w$.
4. Otherwise, by Lemma 6, $(T, \mathcal{X})$ can be refined into a new tree-cut decomposition satisfying the *Invariant*.

The algorithm either stops when we can correctly report that $\mathbf{tcw}(G) > w$ (step 1 or 3) or when the current tree-cut decomposition has no large leaf. In the latter case, as $(T, \mathcal{X})$ satisfies (*), it holds that $\mathbf{tcw}(T, \mathcal{X}) \leqslant 2 \cdot w$. Observe that each refinement step (step 4) strictly increases the number of non-empty bags (see Lemma 6). It follows that the above steps are repeated at most n times, implying that the running time of the 2-approximation algorithm is $2^{O(w^2 \cdot \log w)} \cdot n^2$. □

4 Open problems

A follow-up question is to improve the running time or the approximation factor of our algorithm. Notice that the parameter dependence $2^{O(w^2 \cdot \log w)}$ of the presented algorithm is based on the fact that the tree-cut width is bounded by a quadratic function of treewidth. As we proved in Theorem 2, there is no hope of improving this upper bound. Therefore for any improvement of the parametric dependence, one should avoid using a dynamic programming algorithm on tree decompositions or significantly improve the running time. Another issue is whether we can improve the quadratic dependence on n to a linear one.

To design an exact FPT-algorithm for tree-cut width, that is constructive and uniform, remains open. The $f(w) \cdot n$-time algorithm as observed in Section 1 depends on the explicit description of the obstruction set for tree-width at most w, thus is non-uniform. A related problem is to obtain an upper or lower bound on the size of an obstruction for tree-cut width at most w. As far as we know, no such bounds are known. Notice that an upper bound on the size of an obstruction for tree-width at most w, or alternatively an algorithm generating all obstructions for tree-cut width at most w, turns the aforementioned machinery into a uniform, constructive FPT-algorithm for tree-cut width. Another route to an exact FPT-algorithm, presumably running in linear time in n, is to use the "set of characteristic sequences" technique, as this was done for other width parameters [3, 4, 15, 23–25]. It appears that both approaches to obtain an exact FPT-algorithm would imply a high parametric dependence. It is an intriguing challenge to design an exact FPT-algorithm for tree-cut width whose dependence on n is linear while keeping the parametric dependence reasonably low.

Acknowledgement. The authors would like to thank the anonymous reviewers for helpful remarks that improved the presentation of the paper.

References

1. H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.
2. H. L. Bodlaender, P. G. Drange, M. S. Dregi, F. Fomin, D. Lokshtanov, and M. Pilipczuk. A $O(c^k n)$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016.
3. H. L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996.
4. H. L. Bodlaender and D. M. Thilikos. Computing small search numbers in linear time. In *International Workshop on Parameterized and Exact Computation, IWPEC*, volume 3162 of *Lecture Notes in Computer Science*, pages 37–48. 2004.
5. B. Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.
6. E. D. Demaine and M. T. Hajiaghayi. Bidimensionality: new connections between FPT algorithms and ptass. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 590–601, 2005.

7. M. Dom, D. Lokshtanov, S. Saurabh, and Y. Villanger. Capacitated domination and covering: A parameterized perspective. In *International Workshop on Parameterized and Exact Computation, IWPEC*, volume 5018 of *Lecture Notes in Computer Science*, pages 78–90. 2008.
8. M. R. Fellows, F. V. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh, S. Szeider, and C. Thomassen. On the complexity of some colorful problems parameterized by treewidth. *Information and Computation*, 209(2):143–153, 2011.
9. F. V. Fomin, D. Lokshtanov, V. Raman, and S. Saurabh. Bidimensionality and EPTAS. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 748–759, 2011.
10. F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Bidimensionality and kernels. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 503–510, 2010.
11. R. Ganian, E. J. Kim, and S. Szeider. Algorithmic applications of tree-cut width. In *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part II*, pages 348–360, 2015.
12. M. R. Garey and D. S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness*. 1979.
13. P. A. Golovach and D. M. Thilikos. Paths of bounded length and their cuts: Parameterized complexity and algorithms. *Discrete Optimization*, 8(1):72–86, 2011.
14. M. Grohe, K. Kawarabayashi, D. Marx, and P. Wollan. Finding topological subgraphs is fixed-parameter tractable. In *ACM Symposium on Theory of Computing, STOC*, pages 479–488, 2011.
15. J. Jeong, E. J. Kim, and S. Oum. Constructive algorithm for path-width of matroids. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1695–1704, 2016.
16. E. Kim, S. Oum, C. Paul, I. Sau, and D. M. Thilikos. An FPT 2-approximation for tree-cut decomposition. In *Approximation and Online Algorithms - 13th International Workshop, WAOA 2015, Patras, Greece, September 17-18, 2015. Revised Selected Papers*, pages 35–46, 2015.
17. T. Kloks. *Treewidth. Computations and Approximations*, volume 842 of *LNCS*. Springer, 1994.
18. N. Robertson and P. D. Seymour. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984.
19. N. Robertson and P. D. Seymour. Graph minors. V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(1):92–114, 1986.
20. N. Robertson and P. D. Seymour. Graph minors. XXIII. Nash-Williams' immersion conjecture. *Journal of Combinatorial Theory, Series B*, 100(2):181–205, 2010.
21. P. D. Seymour and R. Thomas. Graph searching and a min-max theorem for tree-width. *Journal of Combinatorial Theory, Series B*, 58(1):22–33, 1993.
22. P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.
23. R. P. Soares. Pursuit-evasion, decompositions and convexity on graphs. *PhD thesis, COATI, INRIA/I3S-CNRS/UNS Sophia Antipolis, France and ParGO Research Group, UFC Fortaleza, Brazil*, 2014.
24. D. M. Thilikos, M. J. Serna, and H. L. Bodlaender. Cutwidth I: A linear time fixed parameter algorithm. *Journal of Algorithms*, 56(1):1–24, 2005.

25. D. M. Thilikos, M. J. Serna, and H. L. Bodlaender. Cutwidth II: Algorithms for partial w-trees of bounded degree. *Journal of Algorithms*, 56(1):25–49, 2005.
26. T. Watanabe, S. Taoka, and T. Mashima. Minimum-cost augmentation to 3-edge-connect all specified vertices in a graph. In *1993 IEEE International Symposium on Circuits and Systems, ISCAS 1993, Chicago, Illinois, USA, May 3-6, 1993*, pages 2311–2314, 1993.
27. P. Wollan. The structure of graphs not admitting a fixed immersion. *Journal of Combinatorial Theory, Series B*, 110:47–66, 2015.