Parameterized Complexity of the MINCCA Problem on Graphs of Bounded Decomposability

Didem Gözüpek, Sibel Özkan, Christophe Paul, Ignasi Sau, Mordeshai Shalom

Parameterized complexity of the MINCCA problem on graphs of bounded decomposability*

Didem Gözüpek[1], Sibel Özkan[2], Christophe Paul[3], Ignasi Sau[3], and Mordechai Shalom[4,5]**

[1] Department of Computer Engineering, Gebze Technical University, Kocaeli, Turkey
didem.gozupek@gtu.edu.tr
[2] Department of Mathematics, Gebze Technical University, Kocaeli, Turkey
s.ozkan@gtu.edu.tr
[3] CNRS, LIRMM, Université de Montpellier, Montpellier, France
paul@lirmm.fr, sau@lirmm.fr
[4] TelHai College, Upper Galilee, 12210, Israel
cmshalom@telhai.ac.il
[5] Department of Industrial Engineering, Boğaziçi University, Istanbul, Turkey

Abstract. In an edge-colored graph, the cost incurred at a vertex on a path when two incident edges with different colors are traversed is called reload or changeover cost. The *Minimum Changeover Cost Arborescence* (MinCCA) problem consists in finding an arborescence with a given root vertex such that the total changeover cost of the internal vertices is minimized. It has been recently proved by Gözüpek *et al.* [14] that the MinCCA problem is FPT when parameterized by the treewidth and the maximum degree of the input graph. In this article we present the following results for MinCCA:

- the problem is W[1]-hard parameterized by the treedepth of the input graph, even on graphs of average degree at most 8. In particular, it is W[1]-hard parameterized by the treewidth of the input graph, which answers the main open problem of [14];
- it is W[1]-hard on multigraphs parameterized by the tree-cut width of the input multigraph;
- it is FPT parameterized by the star tree-cut width of the input graph, which is a slightly restricted version of tree-cut width. This result strictly generalizes the FPT result given in [14];
- it remains NP-hard on planar graphs even when restricted to instances with at most 6 colors and 0/1 symmetric costs, or when restricted to instances with at most 8 colors, maximum degree bounded by 4, and 0/1 symmetric costs.

Keywords: minimum changeover cost arborescence; parameterized complexity; FPT algorithm; treewidth; dynamic programming; planar graph.

1 Introduction

The cost that occurs at a vertex when two incident edges with different colors are crossed over is referred to as *reload cost* or *changeover cost* in the literature. This cost depends on the colors of the traversed edges. Although the reload cost concept has

* This work is supported by the bilateral research program of CNRS and TUBITAK under grant no.114E731.
** The work of this author is supported in part by the TUBITAK 2221 Programme.

important applications in numerous areas such as transportation networks, energy distribution networks, and cognitive radio networks, it has received little attention in the literature. In particular, reload/changeover cost problems have been investigated very little from the perspective of parameterized complexity; the only previous work we are aware of is the one in [14].

In heterogeneous networks in telecommunications, transiting from a technology such as 3G (third generation) to another technology such as wireless local area network (WLAN) has an overhead in terms of delay, power consumption etc., depending on the particular setting. This cost has gained increasing importance due to the recently popular concept of vertical handover [6], which is a technique that allows a mobile user to stay connected to the Internet (without a connection loss) by switching to a different wireless network when necessary. Likewise, switching between different service providers even if they have the same technology has a non-negligible cost. Recently, cognitive radio networks (CRN) have gained increasing attention in the communication networks research community. Unlike other wireless technologies, CRNs are envisioned to operate in a wide range of frequencies. Therefore, switching from one frequency band to another frequency band in a CRN has a significant cost in terms of delay and power consumption [2, 13]. This concept has applications in other areas as well. For instance, the cost of transferring cargo from one mode of transportation to another has a significant cost that outweighs even the cost of transporting the cargo from one place to another using a single mode of transportation [19]. In energy distribution networks, transferring energy from one type of carrier to another has an important cost corresponding to reload costs [8].

The reload cost concept was introduced in [19], where the considered problem is to find a spanning tree having minimum diameter with respect to reload cost. In particular, they proved that the problem cannot be approximated within a factor better than 3 even on graphs with maximum degree 5, in addition to providing a polynomial-time algorithm for graphs with maximum degree 3. The work in [8] extended these inapproximability results by proving that the problem is inapproximable within a factor better than 2 even on graphs with maximum degree 4. When reload costs satisfy the triangle inequality, they showed that the problem is inapproximable within any factor better than 5/3.

The work in [10] focused on the minimum reload cost cycle cover problem, which is to find a set of vertex-disjoint cycles spanning all vertices with minimum total reload cost. They showed an impossibility result for the case when there are 2 colors, the reload costs are symmetric and satisfy the triangle inequality. They also presented some integer programming formulations and computational results.

The authors in [12] study the problems of finding a path, trail or walk connecting two given vertices with minimum total reload cost. They present several polynomial and NP-hard cases for (a)symmetric reload costs and reload costs with(out) triangle inequality. Furthermore, they show that the problem is polynomial for walks, as previously mentioned by [19], and re-proved later for directed graphs by [1].

The work in [9] introduced the *Minimum Changeover Cost Arborescence* (MinCCA) problem. Given a root vertex, MinCCA problem is to find an arborescence with minimum total changeover cost starting from the root vertex. They proved that even on graphs with bounded degree and reload costs adhering to the triangle inequality, MinCCA on directed graphs is inapproximable within $\beta \log\log(n)$ for $\beta > 0$ when there are two colors, and within $n^{1/3-\epsilon}$ for any $\epsilon > 0$ when there are three colors. The work in [15] investigated several special cases of the problem such as bounded cost values, bounded degree, and bounded number of colors. In addition, [15] presented in-

approximability results as well as a polynomial-time algorithm and an approximation algorithm for the considered special cases.

In this paper, we study the MinCCA problem from the perspective of parameterized complexity; see [3, 5, 7, 17]. Unlike the classical complexity theory, parameterized complexity theory takes into account not only the total input size n, but also other aspects of the problem encoded in a parameter k. It mainly aims to find an exact resolution of NP-complete problems. A problem is called *fixed parameter tractable* (FPT) if it can be solved in time $f(k) \cdot p(n)$, where $f(k)$ is a function depending solely on k and $p(n)$ is a polynomial in n. An algorithm constituting such a solution is called an FPT algorithm for the problem. Analogously to NP-completeness in classical complexity, the theory of W[1]-hardness can be used to show that a problem is unlikely to be FPT, i.e., for every algorithm the parameter has to appear in the exponent of n. The parameterized complexity of reload cost problems is largely unexplored in the literature. To the best of our knowledge, [14] is the only work that focuses on this issue by studying the MinCCA problem on bounded treewidth graphs. In particular, [14] showed that the MinCCA problem is in XP when parameterized by the treewidth of the input graph and it is FPT when parameterized by the treewidth *and* the maximum degree of the input graph. In this paper, we prove that the MinCCA problem is W[1]-hard parameterized by the treedepth of the input graph, even on graphs of average degree at most 8. In particular, it is W[1]-hard parameterized by the treewidth of the input graph, which answers the main open issue pointed out by [14]. Furthermore, we prove that it is W[1]-hard on multigraphs parameterized by the tree-cut width of the input multigraph. On the positive side, we present an FPT algorithm parameterized by the star tree-cut width of the input graph, which is a slightly restricted version of tree-cut width that we introduce here. This algorithm strictly generalizes the FPT algorithm given in [14]. We also prove that the problem is NP-hard on planar graphs, which are also graphs of bounded decomposability, even when restricted to instances with at most 6 colors and 0/1 symmetric costs. In addition, we prove that it remains NP-hard on planar graphs even when restricted to instances with at most 8 colors, maximum degree bounded by 4, and 0/1 symmetric costs.

The rest of this paper is organized as follows. In Section 2 we introduce some basic definitions and preliminaries as well as a formal definition of the MinCCA problem. We present our hardness results in Section 3. Due to space limitations, our algorithmic results with respect to star tree-cut width have been completely moved to Appendix D. Finally, Section 4 concludes the paper. The proofs of the results marked with '[⋆]' have been moved to the appendices. All figures except Fig. 1 can be found ni Appendix E.

2 Preliminaries

We say that two partial functions f and f' *agree* if they have the same value everywhere they are both defined, and we denote it by $f \sim f'$. For a set A and an element x, we use $A + x$ (resp., $A - x$) as a shorthand for $A \cup \{x\}$ (resp., $A \setminus \{x\}$). We denote by $[i, k]$ the set of all integers between i and k inclusive, and $[k] = [1, k]$.

Graphs, digraphs, trees, and forests. Given an undirected (multi)graph $G = (V(G), E(G))$ and a subset $U \subseteq V(G)$ of the vertices of G, $\delta_G(U) := \{uu' \in E(G) \mid u \in U, u' \notin U\}$ is the *cut* of G determined by U, i.e., the set of edges of G that have exactly one end in U. In particular, $\delta_G(v)$ denotes the set of edges incident to v in G, and $d_G(v) := |\delta_G(v)|$

is the *degree* of v in G. The *minimum and maximum degrees* of G are defined as $\delta(G) := \min\{d_G(v) \mid v \in V(G)\}$ and $\Delta(G) := \max\{d_G(v) \mid v \in V(G)\}$ respectively. We denote by $N_G(U)$ (resp., $N_G[U]$) the *open* (resp., *closed*) *neighborhood* of U in G. $N_G(U)$ is the set of vertices of $V(G) \setminus U$ that are adjacent to a vertex of U, and $N_G[U] := N_G(U) \cup U$. When there is no ambiguity about the graph G we omit it from the subscripts. For a subset of vertices $U \subseteq V(G)$, $G[U]$ denotes the subgraph of G *induced* by U.

A digraph T is a *rooted tree* or *arborescence* if its underlying graph is a tree and it contains a *root* vertex denoted by $\mathsf{root}(T)$ with a directed path from every other vertex to it. Every other vertex $v \neq \mathsf{root}(T)$ has a parent in T, and v is a *child* of its parent.

A rooted forest is the disjoint union of rooted trees, that is, each connected component of it has a root, which will be called a *sink* of the forest.

Tree decompositions, treewidth, and treedepth. A *tree decomposition* of a graph $G = (V(G), E(G))$ is a tree $\mathcal{T}$, where $V(\mathcal{T}) = \{B_1, B_2, \ldots\}$ is a set of subsets (called *bags*) of $V(G)$ such that the following three conditions are met:

1. $\bigcup V(\mathcal{T}) = V(G)$.
2. For every edge $uv \in E(G)$, $u, v \in B_i$ for some bag $B_i \in V(\mathcal{T})$.
3. For every $B_i, B_j, B_k \in V(\mathcal{T})$ such that B_k is on the path $P_{\mathcal{T}}(B_i, B_j)$, $B_i \cap B_j \subseteq B_k$.

The *width* $\omega(\mathcal{T})$ of a tree decomposition $\mathcal{T}$ is defined as the size of its largest bag minus 1, i.e., $\omega(\mathcal{T}) = \max\{|B| \mid B \in V(\mathcal{T})\} - 1$. The *treewidth* of a graph G, denoted as $\mathbf{tw}(G)$, is defined as the minimum width among all tree decompositions of G. When the treewidth of the input graph is bounded, many efficient algorithms are known for problems that are in general NP-hard. In fact, most problems are known to be FPT when parameterized by the treewidth of the input graph. Hence, what we prove in this paper, i.e. the MinCCA problem is W[1]-hard when parameterized by treewidth, is an interesting result.

The *treedepth* $\mathbf{td}(G)$ of a graph G is the smallest natural number k such that each vertex of G can be labeled with an element from $\{1, \ldots, k\}$ so that every path joining two vertices with the same label contains a vertex having a larger label. Intuitively, where the treewidth parameter measures how far a graph is from being a tree, treedepth parameter measures how far a graph is from being a star. The treewidth of a graph is at most one less than its treedepth; therefore, a W[1]-hardness result for treedepth implies a W[1]-hardness for treewidth.

Tree-cut width. We now explain the concept of tree-cut width and follow the notation in [11]. A *tree-cut decomposition* of a graph G is a pair $(T, \mathcal{X})$ where T is a rooted tree and $\mathcal{X}$ is a partition of $V(G)$ where each set X_t of the partition is associated with a vertex t of T. That is, $\mathcal{X} = \{X_t \subseteq V(G) : t \in V(T)\}$. The set X_t is termed the *bag* associated with the *node* t. For a node t of T we denote by Y_t the union of all the bags associated with t and its descendants, and $G_t = G[Y_t]$. $\mathsf{cut}(t) = \delta(Y_t)$ is the set of all edges with exactly one endpoint in Y_t.

The *adhesion* $\mathsf{adh}(t)$ of t is $|\mathsf{cut}(t)|$. The *torso* of t is the graph H_t obtained from G as follows. Let $t_1, \ldots, t_\ell$ be the children of t, $Y_i = Y_{t_i}$ for $i \in [\ell]$ and $Y_0 = V(G) \setminus (X_t \cup_{i=1}^{\ell} Y_i)$. We first contract each set Y_i to a single vertex y_i, by possibly creating parallel edges. We then remove every vertex y_i of degree 1 (with its incident edge), and finally *suppress* every vertex y_i of degree 2 having 2 neighbors, by connecting its two neighbors with an edge and removing y_i. The *torso size* $\mathsf{tor}(t)$

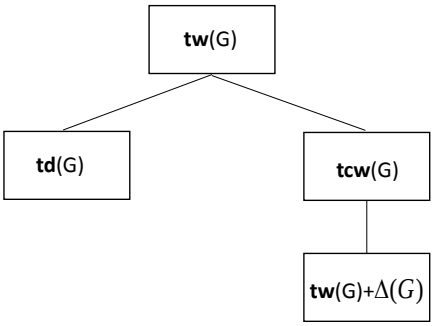

Fig. 1. Relationships between selected graph parameters. A being a child of B means that every graph class with bounded A is also with bounded B but the converse is not necessarily true [11].

of t is the number of vertices in H_t. The *width* of a tree-cut decomposition $(T, \mathcal{X})$ of G is $\max_{t \in V(T)}\{\mathsf{adh}(t), \mathsf{tor}(t)\}$. 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.

Theorem 1 (Kim *et al.* [16]). *Given a graph G on n vertices, a tree-cut decomposition of G of width at most $2\mathbf{tcw}(G)$ can be computed in time $2^{O(k^2 \log \mathbf{tcw}(G))} \cdot n^2$.*

A non-root node t of T is *thin* if $\mathsf{adh}(t) \leq 2$ and *bold* otherwise. A tree-cut decomposition $(T, \mathcal{X})$ is *nice* if for every thin node t and every sibling t' of t, Y_t does not have neighbours in $Y_{t'}$. For a node t of T we use B_t to denote the set of thin children t' of t such that $N(Y_{t'}) \subseteq X_t$, and we let A_t contain every child of t which is not in B_t.

Theorem 2 (Ganian *et al.* [11]). *A tree-cut decomposition of a graph G can be transformed in time $O(n^3)$ to a nice tree-cut decomposition of G without increasing its width or its number of nodes.*

Lemma 1 (Ganian *et al.* [11]). *For every node t of a tree-cut decomposition of width k,*

$$|A_t| \leq 2k + 1. \tag{1}$$

Fig. 1 shows the relationship between selected graph parameters. As depicted in Fig. 1, tree-cut width provides an intermediate measurement which allows either to push the boundary of fixed parameter tractability or strengthen W[1]-hardness result (cf. [11, 16, 20]). Furthermore, Fig. 1 also shows that treedepth and tree-cut width do not possess an implication relation in terms of parameterized complexity.

Reload and changeover costs. We follow the notation and terminology of [19] where the concept of reload cost was defined. We consider edge colored graphs G, where the colors are taken from a finite set X and $\chi : E(G) \to X$ is the *coloring*

function. Given a coloring function χ, we denote by E_x^χ, or simply by E_x the set of edges of E colored x, and $G_x = (V(G), E(G)_x)$ is the subgraph of G having the same vertex set as G, but only the edges colored x. The costs are given by a non-negative function $cc : X^2 \to \mathbb{N}_0$ satisfying

1. $cc(x_1, x_2) = cc(x_2, x_1)$ for every $x_1, x_2 \in X$.
2. $cc(x, x) = 0$ for every $x \in X$.

At this point we note that our algorithms does not rely on any one of the above properties of the cost function, and they will work on other cost functions too. We assume without loss of generality that the minimum non-zero cost value is 1 and we denote the maximum cost value as $C_{max} := \max_{x_1,x_2 \in X} cc(x_1, x_2)$. The cost of traversing two incident edges e_1, e_2 is $cc(e_1, e_2) := cc(\chi(e_1), \chi(e_2))$.

We say that an instance satisfies the *triangle inequality*, if (in addition to the above) the cost function satisfies
$cc(e_1, e_3) \leq cc(e_1, e_2) + cc(e_2, e_3)$ whenever e_1, e_2 and e_3 are incident to the same vertex.

The changeover cost of a path $P = (e_1 - e_2 - \ldots - e_\ell)$ of length ℓ is $cc(P) := \sum_{i=2}^{\ell} cc(e_{i-1}, e_i)$. Note that $cc(P) = 0$ whenever $\ell \leq 1$.

We extend this definition to trees as follows: Given a directed tree T rooted at r, (resp., an undirected tree T and a vertex $r \in V(T)$), for every outgoing edge e of r (resp., incident to r) we define $prev(e) = e$, and for every other edge $prev(e)$ is the edge preceding e on the path from r to e. The changeover cost of T with respect to r is $cc(T, r) := \sum_{e \in E(T)} cc(prev(e), e)$. When there is no ambiguity about the vertex r, we denote $cc(T, r)$ by $cc(T)$.

Statement of the problem. The MinCCA problem aims to find a spanning tree rooted at r with minimum changeover cost [9]. Formally,

MinCCA
Input: A graph $G = (V, E)$ with an edge coloring function $\chi : E \to X$, a vertex $r \in V$ and a changeover cost function $cc : X^2 \to \mathbb{N}_0$.
Output: A spanning tree T of G minimizing $cc(T, r)$.

3 Hardness results

In this section we prove several hardness results for the MinCCA problem. Our main result is in Subsection 3.1, where we prove that the problem is W[1]-hard parameterized by the treedepth of the input graph. We also prove that the problem is W[1]-hard on *multigraphs* parameterized by the tree-cut width of the input graph. Both results hold even if the input graph has bounded average degree. Finally, in Subsection 3.2 we prove that the problem remains NP-hard on planar graphs.

3.1 W[1]-hardness with parameters treedepth and tree-cut width

We need to define the following parameterized problem.

MULTICOLORED k-CLIQUE
Input: A graph G, a coloring function $c : V(G) \to \{1, \ldots, k\}$, and a positive integer k.
Parameter: k.
Question: Does G contain a clique on k vertices with one vertex from each color class?

MULTICOLORED k-CLIQUE is known to be W[1]-hard on general graphs, even in the special case where all color classes have the same number of vertices [18], and therefore we may make this assumption as well.

Theorem 3. *The* MINCCA *problem is* W[1]*-hard parameterized by the treedepth of the input graph, even on graphs with average degree at most 8.*

Proof: We reduce from MULTICOLORED k-CLIQUE, where we may assume that k is odd. Indeed, given an instance (G, c, k) of MULTICOLORED k-CLIQUE, we can trivially reduce the problem to itself as follows. If k is odd, we do nothing. Otherwise, we output $(G', c', k+1)$, where G' is obtained from G by adding a universal vertex v, and $c' : V(G') \to \{1, \ldots, k+1\}$ is such that its restriction to G equals c, and $c(v) = k+1$.

Given an instance (G, c, k) of MULTICOLORED k-CLIQUE with k odd, we proceed to construct an instance (H, X, χ, r, cc) of MINCCA. Let $V(G) = V_1 \uplus V_2 \uplus \cdots \uplus V_k$, where the vertices of V_i are colored i for $1 \leq i \leq k$. Let W be an arbitrary Eulerian circuit of the complete graph K_k, which exists since k is odd. If $V(K_k) = \{v_1, \ldots, v_k\}$, we can clearly assume without loss of generality[6] that W starts by visiting, in this order, vertices $v_1, v_2, \ldots, v_k, v_1$, and that the last edge of W is $\{v_3, v_1\}$; see Fig. 2 and Fig. 3. For every edge $\{v_i, v_j\}$ of W, we add to H a vertex $s_{i,j}$. These vertices are called the *selector vertices* of H. For every two consecutive edges $\{v_i, v_j\}, \{v_j, v_\ell\}$ of W, we add to H a vertex $v_j^{i,\ell}$ and we make it adjacent to both $s_{i,j}$ and $s_{j,\ell}$. We also add to H a new vertex $v_1^{0,2}$ adjacent to $s_{1,2}$, a new vertex $v_1^{3,0}$ adjacent to $s_{3,1}$, and a new vertex r adjacent to $v_1^{0,2}$, which will be the *root* of H. Note that the graph constructed so far is a simple path P on $2\binom{k}{2} + 2$ vertices; see Fig. 4. We say that the vertices of the form $v_j^{i,\ell}$ are *occurrences* of vertex $v_j \in V(K_k)$. For $2 \leq j \leq k$, we add an edge between the root r and the first occurrence of vertex v_j in P (note that the edge between r and the first occurrence of v_1 already exists).

The first k selector vertices, namely $s_{1,2}, s_{2,3}, \ldots, s_{k-1,k}, s_{k,1}$ will play a special role that will become clear later. To this end, for $1 \leq i \leq k$, we add an edge between the selector vertex $s_{i,i \pmod k+1}$ and each of the occurrences of v_i that appear after $s_{i,i \pmod k+1}$ in P. These edges will be called the *jumping edges* of H.

Let us denote by F the graph constructed so far; see Fig. 5. Finally, in order to construct H, we replace each vertex of the form $v_j^{i,\ell}$ in F with a whole copy of the vertex set V_j of G and make each of these new vertices adjacent to all the neighbors of $v_j^{i,\ell}$ in F. This completes the construction of H; see Fig. 6. Note that $\mathbf{td}(H) \leq \binom{k}{2} + 1$, as the removal of the $\binom{k}{2}$ selector vertices from H results in a star centered at r and isolated vertices.

We now proceed to describe the color palette X, the coloring function χ, and the cost function cc, which altogether will encode the edges of G and will ensure the desired properties of the reduction. For simplicity, we associate a distinct color with

[6] This assumption is not crucial for the construction, but helps in making it conceptually and notationally easier.

each edge of H, and thus, with slight abuse of notation, it is enough to describe the cost function cc for every ordered pair of incident edges of H. We will use just three different costs: 0, 1, and B, where B can be set as any real number strictly greater than $\binom{k}{2}$. For each ordered pair of incident edges e_1, e_2 of H, we define

$$cc(e_1,e_2)=\begin{cases} 0, \text{ if } e_1=\{\hat{x},s_{i,j}\} \text{ and } e_2=\{s_{i,j},\hat{y}\} \text{ is a jumping edge such that} \\ \quad \hat{x},\hat{y} \text{ are copies of vertices } x,y\in V_i, \text{ respectively, with } x\neq y, \text{ or} \\ \quad \text{if } e_1=\{r,\hat{x}\} \text{ and } e_2=\{\hat{x},s_{1,2}\}, \text{ where } \hat{x} \text{ is a copy of a vertex} \\ \quad x\in V_1, \text{ or} \\ \quad \text{if } e_1 \text{ and } e_2 \text{ are the two edges that connect a vertex in a copy} \\ \quad \text{of a color class } V_i \text{ to a selector vertex.} \\ 1, \text{ if } e_1=\{\hat{x},s_{i,j}\} \text{ and } e_2=\{s_{i,j},\hat{y}\}, \text{ where } \hat{x} \text{ is a copy of a vertex} \\ \quad x\in V_i \text{ and } \hat{y} \text{ is a copy of a vertex } y\in V_j \text{ such that } \{x,y\}\in E(G). \\ B, \text{ otherwise.} \end{cases}$$

This completes the construction of (H, X, χ, r, cc), which can be clearly performed in polynomial time.

Claim 1. [⋆] *The average degree of H is bounded by* 8.

We now claim that H contains and arborescence T rooted at r with cost at most $\binom{k}{2}$ if and only if G contains a multicolored k-clique[7]. Note that the simple path P described above naturally defines a partial left-to-right ordering among the vertices of H, and hence any arborescence rooted at r contains *forward* and *backward* edges defined in an unambiguous way. Note also that all costs that involve a backward edge are equal to B, and therefore no such edge can be contained in an arborescence of cost at most $\binom{k}{2}$.

Suppose first that G contains a multicolored k-clique with vertices $v_1, v_2, \ldots, v_k$, where $v_i \in V_i$ for $1 \leq i \leq k$. Then we define the edges of the spanning tree T of H as follows. Tree T contains the edges of a left-to-right path Q that starts at the root r, contains all $\binom{k}{2}$ selector vertices and connects them, in each occurrence of a set V_i, to the copy of vertex v_i defined by the k-clique. Since in Q the selector vertices connect copies of pairwise adjacent vertices of G, the cost incurred so far by T is exactly $\binom{k}{2}$. For $1 \leq i \leq k$, we add to Q the edges from r to all vertices in the first occurrence of V_i that are not contained in Q. Note that the addition of these edges to T incurs no additional cost. Finally, we will use the jumping edges to reach the uncovered vertices of H. Namely, for $1 \leq i \leq k$, we add to T an edge between the selector vertex $s_{i,i \pmod k+1}$ and all occurrences of the vertices in V_i distinct from v_i that appear after $s_{i,i \pmod k+1}$; see the green tree in Fig. 6. Note that since the jumping edges in T contain copies of vertices distinct from the the ones in the k-clique, these edges incur no additional cost either. Therefore, $cc(T, r) = \binom{k}{2}$, as we wanted to prove.

Conversely, suppose now that H has an arborescence T rooted at r with cost at most $\binom{k}{2}$. Clearly, all costs incurred by the edges in T are either 0 or 1. For a selector vertex $s_{i,j}$, we call the edges joining $s_{i,j}$ to the vertices in the occurrence of V_i right before $s_{i,j}$ (resp., in the occurrence of V_j right after $s_{i,j}$) the *left* (resp., *right*) edges of this selector vertex.

[7] If the costs associated with colors are restricted to be strictly positive, we can just replace cost 0 with cost ε, for an arbitrarily small positive real number ε, and ask for an arborescence in H of cost strictly smaller than $\binom{k}{2} + 1$.

Claim 2. *Tree T contains exactly one left edge and exactly one right edge of each selector vertex of H.*

Proof: Since only forward edges are allowed in T, and T should be a tree, clearly for each selector vertex exactly one of its left edges belongs to T. Thus, it just remains to prove that T contains exactly one right edge of each selector vertex.

Let $s_{i,j}$ and $s_{j,\ell}$ be two consecutive selector vertices. Let e be the left edge of $s_{j,\ell}$ in T and let v_j be the vertex of the copy of V_j contained in e. Again, since backward edges are not allowed in T, v_j needs to be incident with another forward edge e' of T. If this edge e' contains r or if it is a jumping edge, then the cost incurred in T by e' and e would be equal to B, a contradiction to the hypothesis that $cc(T, r) \leq \binom{k}{2}$. Therefore, e' is necessarily one of the right edges of $s_{i,j}$, so at least one of the right edges of selector vertex $s_{i,j}$ belongs to T.

As for the right edges of the last selector vertex, namely $s_{3,1}$, if none of them belonged to T, then there would be a jumping edge going to the last copy of V_1 such that, together with the left edge of selector vertex $s_{1,2}$ that belongs to T, would incur a cost of B, which is impossible.

We have already proved that exactly one left edge and *at least* one of the right edges of each selector vertex belong to T. For each selector vertex $s_{i,j}$, its left edge in T together with *each* of its right edges in T incur at cost of at least 1. But as there are $\binom{k}{2}$ selector vertices in H, and by hypothesis the cost of T is at most $\binom{k}{2}$, we conclude that exactly one of the right edges of each selector vertex belongs to T, as we wanted to prove. ◊

By Claim 2, tree T contains a path Q' that chooses exactly one vertex from each occurrence of a color class of G. We shall now prove that, thanks to the jumping edges, these choices are *coherent*, which will allow us to extract the desired multicolored k-clique in G.

Claim 3. *For every $1 \leq i \leq k$, the vertices in the copies of color class V_i contained in Q' all correspond to the same vertex of G, denoted by v_i.*

Proof: Assume for contradiction that for some index i, the vertices in the copies of color class V_i contained in Q' correspond to at least two distinct vertices v_i and v_i' of G, in such a way that v_i is the selected vertex in the first occurrence of V_i, and v_i' occurs later, say in the jth occurrence of V_i. Therefore, the copy of v_i in the jth occurrence of V_i does not belong to path Q', so for this vertex to be contained in T, by construction it is necessarily an endpoint of a jumping edge e starting at the selector vertex $s_{i,i \pmod k+1}$. But then the cost incurred in T by the edges e' and e, where e' is the edge joining the copy of v_i in the first occurrence of V_i to the selector vertex $s_{i,i \pmod k+1}$, equals B, contradicting the hypothesis that $cc(T, r) \leq \binom{k}{2}$. ◊

Finally, we claim that the vertices $v_1, v_2, \ldots, v_k$ defined by Claim 3 induce a multicolored k-clique in G. Indeed, assume for contradiction that there exist two such vertices v_i and v_j such that $\{v_i, v_j\} \notin E(G)$. Then the cost in T incurred by the two edges connecting the copies of v_i and v_j to the selector vertex $s_{i,j}$ (by Claim 2, these two edges indeed belong to T) would be equal to B, contracting again the hypothesis that $cc(T, r) \leq \binom{k}{2}$. This concludes the proof of the theorem. □

In the next theorem we prove that the MinCCA problem is W[1]-hard on multigraphs parameterized by the tree-cut width of the input graph. Note that this result does not imply Theorem 3, which applies to graphs without multiple edges.

Theorem 4. [⋆] *The* MinCCA *problem is* W[1]*-hard on multigraphs parameterized by the tree-cut width of the input multigraph.*

3.2 NP-hardness on planar graphs

In this subsection we prove that the MinCCA problem remains NP-hard on planar graphs. In order to prove this result, we need to introduce the Planar Monotone 3-sat problem. An instance of 3-sat is called *monotone* if each clause is monotone, that is, each clause consists only of positive variables or only of negative variables. We call a clause with only positive (resp., negative) variables a *positive* (resp., *negative*) clause. Given an instance ϕ of 3-sat, we define the bipartite graph G_ϕ that has one vertex per each variable and each clause, and has an edge between a variable-vertex and a clause-vertex if and only if the variable appears (positively or negatively) in the clause. A *monotone rectilinear representation* of a monotone 3-sat instance ϕ is a *planar* drawing of G_ϕ such that all variable-vertices lie on a path, all positive clause-vertices lie *above* the path, and all negative clause-vertices lie *below* the path (see Fig. 7). In the Planar Monotone 3-sat problem, we are given a monotone rectilinear representation of a planar monotone 3-sat instance ϕ, and the objective is to determine whether ϕ is satisfiable. Berg and Khosravi [4] proved that the Planar Monotone 3-sat problem is NP-complete.

Theorem 5. *The* MinCCA *problem is* NP*-hard on planar graphs even when restricted to instances with at most 6 colors and 0/1 symmetric costs.*

Proof: We reduce from the Planar Monotone 3-sat problem. Given a monotone rectilinear representation of a planar monotone 3-sat instance ϕ, we build an instance (H, X, χ, r, f) of MinCCA as follows. We denote the variable-vertices of G_ϕ as $\{x_1, \ldots, x_n\}$ and the clause-vertices of G_ϕ as $\{C_1, \ldots, C_m\}$. Without loss of generality, we assume that the variable-vertices appear in the order $x_1, \ldots, x_n$ on the path P of G_ϕ that links the variable-vertices. For every variable-vertex x_i of G_ϕ, we add to H a gadget consisting of four vertices $x_i^\ell, x_i^r, x_i^+, x_i^-$ and five edges $\{x_i^\ell, x_i^+\}$, $\{x_i^+, x_i^r\}$, $\{x_i^r, x_i^-\}$, $\{x_i^-, x_i^\ell\}$, $\{x_i^+, x_i^-\}$. We add to H a new vertex r, which we set as the root, and we add the edge $\{r, x_1^\ell\}$. For every $i \in \{1, \ldots, n-1\}$, we add to H the edge $\{x_i^r, x_{i+1}^\ell\}$. We add to H all clause-vertices $C_1, \ldots, C_m$. For every $i \in \{1, \ldots, n\}$, we add an edge between vertex x_i^+ and each clause-vertex of G_ϕ in which variable x_i appears positively, and an edge between vertex x_i^- and each clause-vertex of G_ϕ in which variable x_i appears negatively. This completes the construction of H, which is illustrated in Fig. 8. Since G_ϕ is planar and all positive (resp., negative) clause-vertices appear above (resp., below) the path P, it is easy to see that the graph H is planar as well.

We define the color palette as $X = \{1, 2, 3, 4, 5, 6\}$. Let us now describe the edge-coloring function χ. For every clause-vertex C_j, we color arbitrarily its three incident edges with the colors $\{4, 5, 6\}$, so that each edge incident to C_j gets a different color. For every $i \in \{1, \ldots, n\}$, we define $\chi(\{x_i^\ell, x_i^+\}) = \chi(\{x_i^r, x_i^-\}) = 1$, $\chi(\{x_i^+, x_i^r\}) = \chi(\{x_i^-, x_i^\ell\}) = 2$, and $\chi(\{x_i^+, x_i^-\}) = 3$. We set $\chi(\{r, x_1^\ell\}) = 4$ and for every $i \in \{1, \ldots, n-1\}$, $\chi(\{x_i^r, x_{i+1}^\ell\}) = 4$. The function χ is also depicted in Fig. 8. Finally, we define the cost function cc to be symmetric and, for every $i \in \{1, 2, 3, 4, 5, 6\}$, we set $cc(i, i) = 0$. We define $cc(1, 2) = 1$ and $cc(1, 3) = cc(2, 3) = 0$. For every $i \in \{4, 5, 6\}$, we set $cc(1, i) = cc(2, i) = 0$ and $cc(3, i) = 1$. Finally, for every $i, j \in \{4, 5, 6\}$ with $i \neq j$ we set $cc(i, j) = 1$.

We now claim that H contains an arborescence T rooted at r with cost 0 if and only if the formula ϕ is satisfiable.

Suppose first that ϕ is satisfiable, and fix a satisfying assignment of ϕ. We proceed to define an arborescence T rooted at T with cost 0. T contains the edge $\{r, x_1^\ell\}$ and, for every $i \in \{1, \ldots, n-1\}$, the edge $\{x_i^r, x_{i+1}^\ell\}$. For every $i \in \{1, \ldots, n\}$, if variable x_i is set to 1 in the satisfying assignment of ϕ, we add to T the edges $\{x_i^\ell, x_i^+\}$, $\{x_i^+, x_i^-\}$, and $\{x_i^-, x_i^r\}$. Otherwise, if variable x_i is set to 0 in the satisfying assignment of ϕ, we add to T the edges $\{x_i^\ell, x_i^-\}$, $\{x_i^-, x_i^+\}$, and $\{x_i^+, x_i^r\}$. Finally, for every $j \in \{1, \ldots, m\}$, let x_t (resp., $\bar{x}_t$) be a literal in clause C_j that is set to 1 by the satisfying assignment of ϕ (note that for each clause we consider only *one* such literal). Then we add to T an edge between vertex C_j and vertex x_t^+ (resp., x_t^-). It can be easily checked that T is an arborescence of H with cost 0.

Conversely, suppose now that H contains an arborescence T rooted at r with cost 0, and let us define a satisfying assignment of ϕ. Since for every $i, j \in \{4, 5, 6\}$ with $i \neq j$ we have that $cc(i, j) = 1$, for every clause-vertex C_j exactly one of its incident edges belongs to T. From the structure of H and from the fact that $cc(1, 2) = 1$, it follows that in order for the tree T to span all vertices of H, for every $i \in \{1, \ldots, n\}$ either the three edges $\{x_i^\ell, x_i^+\}$, $\{x_i^+, x_i^-\}$, $\{x_i^-, x_i^r\}$ or the three edges $\{x_i^\ell, x_i^-\}$, $\{x_i^-, x_i^+\}$, $\{x_i^+, x_i^r\}$ belong to T. In the former case, we set variable x_i to 1, and in the latter case we set variable x_i to 0. Since for every $i \in \{4, 5, 6\}$ we have that $cc(3, i) = 1$, it follows that for every clause-vertex C_j, its incident edge that belongs to T joins C_j to a literal that is set to 1 by the constructed assignment. Hence, all clauses of ϕ are satisfied by this assignment, concluding the proof of the theorem. □

Note that the above proof actually implies that MinCCA cannot be approximated to *any* positive ratio on planar graphs in polynomial time, since an optimal solution has cost 0. We do not know whether such a strong inapproximability result holds even if we do not allow to use costs 0 among different colors.

In the next theorem we present a modification of the previous reduction showing that the MinCCA problem remains hard even if the maximum degree of the input planar graph is bounded.

Theorem 6. [⋆] *The* MinCCA *problem is* NP*-hard on planar graphs even when restricted to instances with at most 8 colors, maximum degree bounded by* 4*, and 0/1 symmetric costs.*

4 Conclusions and further research

In this article we proved several hardness results for the MinCCA problem. In particular, we proved that the problem is W[1]-hard parameterized by treewidth on general graphs, and that it is NP-hard on planar graphs, but we do not know whether it is W[1]-hard parameterized by treewidth (or treedepth) on planar graphs.

On the other hand, we provided an FPT algorithm for a restricted version of tree-cut width, and we proved that the problem is W[1]-hard on multigraphs parameterized by tree-cut width. While we were not able to prove this W[1]-hardness result on graphs without multiple edges, we believe that it is indeed the case.

Finally, it would be interesting to try to generalize our techniques to prove hardness results or to provide efficient algorithms for other reload cost problems that have been studied in the literature [6, 8, 10, 19].

References

1. E. Amaldi, G. Galbiati, and F. Maffioli. On minimum reload cost paths, tours, and flows. *Networks*, 57(3):254–260, 2011.
2. S. Arkoulis, E. Anifantis, V. Karyotis, S. Papavassiliou, and N. Mitrou. On the optimal, fair and channel-aware cognitive radio network reconfiguration. *Computer Networks*, 57(8):1739–1757, 2013.
3. M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.
4. M. de Berg and A. Khosravi. Optimal binary space partitions for segments in the plane. *International Journal of Computational Geometry and Applications*, 22(3):187–206, 2012.
5. R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.
6. M. R. Çelenlioğlu, D. Gözüpek, and H. A. Mantar. A survey on the energy efficiency of vertical handover mechanisms. In *Proc. of the International Conference on Wireless and Mobile Networks (WiMoN)*, 2013.
7. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. Springer, 2006.
8. G. Galbiati. The complexity of a minimum reload cost diameter problem. *Discrete Applied Mathematics*, 156(18):3494–3497, 2008.
9. G. Galbiati, S. Gualandi, and F. Maffioli. On minimum changeover cost arborescences. In *Proc. of the 10th International Symposium on Experimental Algorithms (SEA)*, volume 6630 of *LNCS*, pages 112–123, 2011.
10. G. Galbiati, S. Gualandi, and F. Maffioli. On minimum reload cost cycle cover. *Discrete Applied Mathematics*, 164:112–120, 2014.
11. R. Ganian, E. J. Kim, and S. Szeider. Algorithmic applications of tree-cut width. In *Proc. of the 40th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 9235 of *LNCS*, pages 348–360, 2015.
12. L. Gourvès, A. Lyra, C. Martinhon, and J. Monnot. The minimum reload s-t path, trail and walk problems. *Discrete Applied Mathematics*, 158(13):1404–1417, July 2010.
13. D. Gozupek, S. Buhari, and F. Alagoz. A spectrum switching delay-aware scheduling algorithm for centralized cognitive radio networks. *IEEE Transactions on Mobile Computing*, 12(7):1270–1280, 2013.
14. D. Gözüpek, H. Shachnai, M. Shalom, and S. Zaks. Constructing minimum changeover cost arborescences in bounded treewidth graphs. DOI: 10.1016/j.tcs.2016.01.022, to appear in *Theoretical Computer Science*, 2016.
15. D. Gözüpek, M. Shalom, A. Voloshin, and S. Zaks. On the complexity of constructing minimum changeover cost arborescences. *Theoretical Computer Science*, 540:40–52, 2014.
16. E. J. Kim, S.-I. Oum, C. Paul, I. Sau, and D. M. Thilikos. An FPT 2-Approximation for Tree-cut Decomposition. In *Proc. of the 13th Workshop on Approximation and Online Algorithms (WAOA)*, volume 9499 of *LNCS*, pages 35–46, 2015.
17. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*, volume 31. Oxford University Press, 2006.
18. K. Pietrzak. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *Journal of Computer and System Sciences*, 67(4):757–771, 2003.
19. H.-C. Wirth and J. Steffan. Reload cost problems: minimum diameter spanning tree. *Discrete Applied Mathematics*, 113(1):73–85, 2001.
20. P. Wollan. The structure of graphs not admitting a fixed immersion. *Journal of Combinatorial Theory, Series B*, 110:47–66, 2015.

A Proof of Claim 1

Recall that we assumed that there are k color classes, each with n vertices. Note that there are $\binom{k}{2}$ selector vertices and a root vertex r. Since each vertex of the form $v_j^{i,\ell}$ is replaced with a whole copy of the vertex set V_j, there are $\binom{k}{2}+1n$ additional vertices. Therefore, $|V(H)| = (\binom{k}{2}+1)n+\binom{k}{2}+1$. Moreover, $\binom{k}{2}-k$ selector vertices do not have any jumping edges; the neighborhood of such a selector vertex is exactly the vertices corresponding to its neighbors on the simple path P described previously. Thus, the number of edges incident to these vertices without jumping edges is $(\binom{k}{2}-k)2n$. For each of the first k selector vertices, when we sum up its jumping edges and its edges incident to the n vertices corresponding to its successor vertex on path P, we get at most $(k-1)n$ edges because at most $k-1$ edges incident to the corresponding vertex remain on the Eulerian circuit. Together with its edges incident to the n vertices corresponding to its predecessor vertex on path P, each of the first k selector vertices has k^2n incident edges. Since root vertex r is adjacent to the first occurrence of vertex v_j in P for $1 \leq j \leq k$, kn edges are incident to vertex r. Hence, if we denote by $\overline{\mathsf{deg}}(H)$ the average degree of H, we get:

$$\overline{\mathsf{deg}}(H) = \frac{2|E(H)|}{|V(H)|} \leq \frac{kn + k^2n + (\binom{k}{2}-k)2n}{(\binom{k}{2}+1)n + \binom{k}{2}+1} = \frac{8k(k-1)}{2k^2-3k+4} \leq 8.$$

B Proof of Theorem 4

As in Theorem 3, we reduce again from Multicolored k-Clique. Given an instance (G, c, k) of Multicolored k-Clique with k odd, we proceed to construct an instance (H, X, χ, r, cc) of MinCCA. The first steps of the construction resemble the ones of Theorem 3. Namely, let F be the graph constructed in the proof of Theorem 3, and let F' be the graph constructed from F as follows. We delete the last vertex of F, namely $v_1^{3,0}$, and all edges incident with the root r except the edge $\{r, v_1^{0,2}\}$. Finally, for every vertex of F' of the form $v_\ell^{i,j}$ (that is, a vertex that is neither the root nor a selector vertex), let e_1 and e_2 be the two edges of the path P containing $v_\ell^{i,j}$, such that e_1 is to the left of e_2. Then we contract the edge e_2. This completes the construction of F'. Note that $|V(F')| = \binom{k}{2}+1$. Finally, in order to construct H, we proceed as follows. For every edge e of F' which is not a jumping edge, let $s_{i,j}$ be its right endpoint. Then we replace e with a multiedge with multiplicity $|V_i|$, and we associate each of these edges with a distinct vertex in $V_i \subseteq V(G)$. These edges are called the *horizontal* edges of H. On the other hand, for every $1 \leq i \leq k$, and for every jumping edge e whose left endpoint is the selector vertex $s_{i,i \pmod k+1}$, we replace e with a multiedge with multiplicity $|V_i|$, and we subdivide each of these new edges once. Each of these new vertices is associated with a distinct vertex in $V_i \subseteq V(G)$. Let us call the selector vertices of the form $s_{i,i \pmod k+1}$ *special* selector vertices. This completes the construction of H. Note that, as in the proof of Theorem 3, the average degree of H is also bounded by a constant (multiedges are counted with their multiplicity)

Claim 4. *The tree-cut width of H is at most $\binom{k}{2}+1$.*

Proof: We proceed to construct a tree-cut decomposition $(T, \mathcal{X})$ of H of width at most $\binom{k}{2}+1$. Let T be a star with $|V(H)| - \binom{k}{2} - 1$ leaves rooted at its center. If t is

the center of this star T, then the bag X_t contains the root of H together with the $\binom{k}{2}$ selector vertices. If t is a leaf of T, then the bag X_t contains a single vertex, in such a way that each of the remaining $|V(H)| - \binom{k}{2} - 1$ vertices of T is associated with one of the leaves. For every leaf $v \in V(T)$, it holds that $\mathsf{adh}(t) = 2$, as every vertex in H that is neither the root nor a selector vertex has degree exactly 2. Also, for every leaf t of T, clearly $\mathsf{tor}(t) \leq 2$, as $|X_t| = 1$ and t has degree 1 in T. Finally, if t the root of T, then when considering the torso H_t, every vertex in a leaf-bag gets dissolved, as each such vertex has exactly 2 neighbors in X_t. Therefore, $\mathsf{tor}(t) = \binom{k}{2} + 1$. $\Diamond$

We now proceed to describe the color palette X, the coloring function χ, and the cost function cc, which altogether will encode the edges of G and will ensure the desired properties of the reduction. For simplicity, as in the proof of Theorem 3, we again associate a distinct color with every edge of H, and thus, it is enough to describe the cost function cc for every ordered pair of incident edges of H. In this case, we will use just two different costs: 0 and 1. For every ordered pair of incident edges e_1, e_2 of H, we define

$$cc(e_1, e_2) = \begin{cases} 0, \text{ if } e_1 = \{x, s_{i,j}\} \text{ and } e_2 = \{s_{i,j}, y\} \text{ are two horizontal edges} \\ \quad \text{containing a selector vertex } s_{i,j}, \text{ such that } x \text{ is to the left} \\ \quad \text{of } y, \text{ and the vertex in } V_i \text{ associated with } e_1 \text{ is adjacent in } G \\ \quad \text{to the vertex in } V_j \text{ associated with } e_2, \text{ or} \\ \quad \text{if } e_1 = \{x, s_{i,i \pmod k+1}\} \text{ and } e_2 = \{s_{i,i \pmod k+1}, \hat{x}'_i\} \text{ are} \\ \quad \text{such that } s_{i,i \pmod k+1} \text{ is a special selector vertex, edge } e_1 \\ \quad \text{is horizontal and is associated with a vertex } v_i \in V_i, \text{ edge } e_2 \\ \quad \text{arises from the subdivision of a jumping edge such that vertex} \\ \quad \hat{x}'_i \text{ is associated with a vertex } v'_i \in V_i \text{ with } v_i \neq v'_i, \text{ or} \\ \quad \text{if } e_1 = \{x, s_{i,j}\} \text{ and } e_2 = \{s_{i,j}, \hat{x}'_i\} \text{ with } s_{i,j} \text{ being a selector} \\ \quad \text{vertex that is not special, edge } e_1 \text{ is horizontal and is associated} \\ \quad \text{with a vertex } v_i \in V_i, \text{ edge } e_2 \text{ arises from the subdivision of a} \\ \quad \text{jumping edge such that vertex } \hat{x}'_i \text{ is associated with a vertex} \\ \quad v'_i \in V_i \text{ with } v_i = v'_i. \\ 1, \text{ otherwise.} \end{cases}$$

The three different cases above where $cc(e_1, e_2) = 1$ are illustrated in Fig 9(a)-(b)-(c), respectively. This completes the construction of (H, X, χ, r, cc), which can be clearly performed in polynomial time. We now claim that H contains and arborescence T rooted at r with cost 0 if and only if G contains a multicolored k-clique. Again, we assume that any arborescence in H rooted at r contains *forward* and *backward* edges defined in an unambiguous way.

Suppose first that G contains a multicolored k-clique with vertices $v_1, v_2, \ldots, v_k$, where $v_i \in V_i$ for $1 \leq i \leq k$. Then we define the edges of the spanning tree T of H as follows. For each selector vertex $s_{i,j}$, $1 \leq i, j \leq k$, we add to T its left horizontal edge associated with the vertex v_i that belongs to the clique. For every jumping edge $\{s_{i,i \pmod k+1}, s_{i,j}\}$ of T', for some $1 \leq j \leq k$ with $j \notin \{i, i \pmod k + 1\}$, we do the following. Note that this edge has given rise to $|V_i|$ paths with two edges in H, and the vertices of H in the middle of these paths, which we call *inner* vertices, are associated with the vertices in V_i. Then we add to H a forward edge between $s_{i,i \pmod k+1}$ and each inner vertex $\hat{x}$ associated with a vertex in V_i distinct from the vertex v_i that belongs to the clique. Note that $|V_i| - 1$ edges are added to H in this way. Finally, we add a backward edge between $s_{i,j}$ and the inner vertex $\hat{x}$ associated with the vertex

$v_i \in V_i$ that belongs to the clique. By the definition of the cost function and using the fact that the vertices $v_1, v_2, \ldots, v_k$ are pairwise adjacent in G, it can be easily checked that $cc(T, r) = 0$, as we wanted to prove.

Conversely, suppose now that H has an arborescence T rooted at r with cost 0. Clearly, all costs incurred by the edges in T are necessarily 0. Since the cost incurred by the two edges incident with every inner vertex is equal to 1, necessarily T contains a path Q starting at the root r and containing all $\binom{k}{2}$ selector vertices. Let $s_{i,j}$ be an arbitrary selector vertex distinct from the last one, and let e_i and e_j be its left and right incident horizontal edges in Q, respectively. Since $cc(e_1, e_2) = 0$, necessarily the vertex $v_i \in V_i$ associated with e_i is adjacent in G to the vertex $v_j \in V_j$ associated with e_j. We say that the selector vertex $s_{i,j}$ has *selected* the vertex v_i. Our objective is to prove that these selections are *coherent*, in the sense that if two distinct selector vertices $s_{i,j}$ and $s_{i,\ell}$ have selected vertices v_i and v'_i in V_i, respectively, then $v_i = v'_i$. This property will be guaranteed by how the inner vertices are covered by H, as we proceed to prove.

By construction of H, each such inner vertex $\hat{x}$ is adjacent to a special selector vertex, say $s_{i,i+1}$, and to another selector vertex that is not special, say $s_{i,j}$. Let $e_1 = \{s_{i,i+1}, \hat{x}\}$ and let $e_2 = \{\hat{x}, s_{i,j}\}$. Note that either e_1 or e_2 belong to T, but not both, as otherwise these two edges would close a cycle with the path Q. Let also e_i (resp., e'_i) be the edge that is to the left of $s_{i,i+1}$ (resp., $s_{i,j}$) in Q; see Fig 9(d) for an illustration. Note that e_i (resp., e'_i) is associated with a vertex $v_i \in V_i$ (resp., $v'_i \in V_i$), and similarly vertex $\hat{x}$ is associated with another vertex $v''_i \in V_i$. We distinguish two cases. Assume first that $v_i = v''_i$. In this case, by the definition of the cost function we have that $cc(e_i, e_1) = 1$, so e_1 cannot belong to T, implying that e_2 belongs to T, which is possible only if $cc(e'_i, e_2) = 1$, and this is true if and only if $v'_i = v_i$. This implies that the selections made by the selection vertices are coherent, as we wanted to prove. Otherwise, we have that $v_i \neq v''_i$. By the definition of the cost function, it holds that $cc(e'_i, e_2) = 1$, and therefore, as we assume that $cc(T, r) = 0$, necessarily e_1 belongs to T, which is indeed possible as $cc(e_i, e_1) = 0$ because $v_i \neq v''_i$.

By the above discussion, it follows that for each $1 \leq i \leq k$, all selector vertices of the form $s_{i,j}$, for every $1 \leq j \leq k$, $i \neq j$, have selected the same vertex $v_i \in V_i$. Furthermore, for every $1 \leq i, j \leq k$, $i \neq j$, it holds that $\{v_i, v_j\} \in E(G)$. That is, the selected vertices $v_1, v_2, \ldots, v_k$ induce a multicolored k-clique in G, concluding the proof of the theorem.

C Proof of Theorem 6

The reduction follows closely the one of Theorem 5. Given a monotone rectilinear representation of a planar monotone 3-SAT instance ϕ, we build an instance (H, X, χ, r, f) of MinCCA as follows. We denote the variable-vertices of G_ϕ as $\{x_1, \ldots, x_n\}$ and the clause-vertices of G_ϕ as $\{C_1, \ldots, C_m\}$. Without loss of generality, we assume that the variable-vertices appear in the order $x_1, \ldots, x_n$. For every variable-vertex x_i of G_ϕ, we add to H a gadget consisting of four vertices $x_i^\ell, x_i^r, x_i^+, x_i^-$ and five edges $\{x_i^\ell, x_i^+\}$, $\{x_i^+, x_i^r\}$, $\{x_i^r, x_i^-\}$, $\{x_i^-, x_i^\ell\}$, $\{x_i^+, x_i^-\}$. We add to H a new vertex r, which we set as the root, and we add the edge $\{r, x_1^\ell\}$. Let $\mathcal{C}_i^+$ be the set of clauses that variable i appears positively and let $\mathcal{C}_i^-$ be the set of clauses that variable i appears negatively. For every $i \in \{1, \ldots, n\}$ and for every clause $j \in \{1, \ldots, |\mathcal{C}_i^+|\}$, we add vertices x_{ij}^+ and x_{ij}^{r+}. Likewise, for every $i \in \{1, \ldots, n\}$ and for every clause $j \in \{1, \ldots, |\mathcal{C}_i^-|\}$, we add vertices x_{ij}^- and x_{ij}^{r-}. Moreover, for every $i \in \{1, \ldots, n-1\}$, we add a vertex x'_i as

well as the edges $\{x_i^r, x_i'\}$ and $\{x_i', x_{i+1}^l\}$. We proceed our construction by adding for every $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, |\mathcal{C}_i^+|\}$ the edges $\{x_i^+, x_{ij}^+\}$, $\{x_i^r, x_{ij}^{r+}\}$, $\{x_{ij}^+, x_{ij}^{r+}\}$, $\{x_{ij}^+, C_j\}$ and for every $j \in \{1, \ldots, |\mathcal{C}_i^-|\}$ the edges $\{x_i^-, x_{ij}^-\}$, $\{x_i', x_{ij}^{r-}\}$, $\{x_{ij}^-, x_{ij}^{r-}\}$, $\{x_{ij}^-, C_j\}$. Subsequently, for every $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, |\mathcal{C}_i^+| - 1\}$ we add the edges $\{x_{ij}^+, x_{i(j+1)}^+\}$, $\{x_{ij}^{r+}, x_{i(j+1)}^{r+}\}$, and for every $j \in \{1, \ldots, |\mathcal{C}_i^-| - 1\}$ we add the edges $\{x_{ij}^-, x_{i(j+1)}^-\}$, $\{x_{ij}^{r-}, x_{i(j+1)}^{r-}\}$. Note that the maximum degree of H is indeed 4. An example of this construction can be found in Fig 10.

We define the color palette as $X = \{1, 2, 3, 4, 5, 6, 7, 8\}$. Let us now describe the edge coloring function χ. For every clause vertex, we arbitrarily color its three incident edges with colors $\{4, 5, 6\}$ so that each incident edge gets a different color. For every $i \in \{1, \ldots, n\}$, we define $\chi(\{x_i^\ell, x_i^+\}) = \chi(\{x_i^r, x_i^-\}) = 1$, $\chi(\{x_i^+, x_i^r\}) = \chi(\{x_i^-, x_i^\ell\}) = 2$, and $\chi(\{x_i^+, x_i^-\}) = 3$. We set $\chi(\{r, x_1^\ell\}) = 4$ and for every $i \in \{1, \ldots, n-1\}$, $\chi(\{x_i^r, x_i'\}) = \chi(x_i', x_{i+1}^l) = 4$. For every $j \in \{1, \ldots, |\mathcal{C}_i^+| - 1\}$, we set $\chi(x_{ij}^+, x_{i(j+1)}^+) = 7$ and $\chi(x_{ij}^{r+}, x_{i(j+1)}^{r+}) = 8$. Likewise, for every $j \in \{1, \ldots, |\mathcal{C}_i^-| - 1\}$, we set $\chi(x_{ij}^-, x_{i(j+1)}^-) = 7$ and $\chi(x_{ij}^{r-}, x_{i(j+1)}^{r-}) = 8$. For every $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, |\mathcal{C}_i^+|\}$, we set $\chi(x_{ij}^+, x_{ij}^{r+}) = \chi(x_i^r, x_{ij}^{r+}) = 8$, $\chi(x_i^+, x_{ij}^+) = 7$ and for every $j \in \{1, \ldots, |\mathcal{C}_i^-|\}$ we set $\chi(x_i', x_{ij}^{r-}) = \chi(x_{ij}^-, x_{ij}^{r-}) = 8$, $\chi(x_i^-, x_{ij}^-) = 7$. The function χ is also depicted in Fig. 10. Finally, we define the cost function cc as follows: for every $i \in \{1, 2, 3, 4, 5, 6, 7, 8\}$, we set $cc(i, i) = 0$, $cc(1, 2) = 1$ and $cc(1, 3) = cc(2, 3) = 0$. For every $i, j \in \{4, 5, 6\}$ with $i \neq j$ we set $cc(i, j) = 1$. For all $i \in \{1, 2, 3, 4, 5, 6\}$, $cc(8, i) = 0$, whereas $cc(1, 8) = cc(2, 8) = 0$ and $cc(7, 8) = cc(8, 7) = 1$. Moreover, $cc(1, 4) = cc(2, 4) = 0$ and for all $i \in \{4, 5, 6\}$ we set $cc(8, i) = cc(i, 8) = 0$ and $cc(7, i) = cc(i, 7) = 1$.

We now claim that H contains an arborescence T rooted at r with cost 0 if and only if the formula ϕ is satisfiable.

Suppose first that ϕ is satisfiable, and we proceed to define an arborescence T rooted at r with cost 0. T contains the edge $\{r, x_1^\ell\}$ and, for every $i \in \{1, \ldots, n-1\}$, the edges $\{x_i^r, x_i'\}$ and $\{x_i', x_{i+1}^\ell\}$. For every $i \in \{1, \ldots, n\}$, if variable x_i is set to 1 in the satisfying assignment of ϕ, we add to T the edges $\{x_i^\ell, x_i^+\}$, $\{x_i^+, x_i^-\}$, and $\{x_i^-, x_i^r\}$. Otherwise, if variable x_i is set to 0 in the satisfying assignment of ϕ, we add to T the edges $\{x_i^\ell, x_i^-\}$, $\{x_i^-, x_i^+\}$, and $\{x_i^+, x_i^r\}$. For every $i \in \{1, \ldots, n\}$, if variable x_i is set to 1 in the satisfying assignment of ϕ, then for every $j \in \{1, \ldots, |C_i^+|\}$ we add to T the edges $\{x_i^r, x_{ij}^{r+}\}$, $\{x_{ij}^{r+}, x_{ij}^+\}$, $\{x_{ij}^+, C_j\}$ (note that for each clause C_j we add only one such edge), for every $j \in \{1, \ldots, |C_i^+| - 1\}$ we add to T the edges $\{x_{ij}^{r+}, x_{i(j+1)}^{r+}\}$, for every $j \in \{1, \ldots, |C_i^-|\}$ we add to T the edges $\{x_i^-, x_{ij}^-\}$, $\{x_i', x_{ij}^{r-}\}$, and for every $j \in \{1, \ldots, |C_i^-| - 1\}$ we add to T the edges $\{x_{ij}^{r-}, x_{i(j+1)}^{r-}\}$, $\{x_{ij}^-, x_{i(j+1)}^-\}$. For every $i \in \{1, \ldots, n\}$, if variable x_i is set to 0 in the satisfying assignment of ϕ, then for every $j \in \{1, \ldots, |C_i^+|\}$ we add to T the edges $\{x_i^r, x_{ij}^{r+}\}$, $\{x_i^+, x_{ij}^+\}$, for every $j \in \{1, \ldots, |C_i^+| - 1\}$ we add to T the edges $\{x_{ij}^{r+}, x_{i(j+1)}^{r+}\}$, $\{x_{ij}^+, x_{i(j+1)}^+\}$, for every $j \in \{1, \ldots, |C_i^-|\}$ we add to T the edges $\{x_i', x_{ij}^{r-}\}$, $\{x_{ij}^{r-}, x_{ij}^-\}$, $\{x_{ij}^-, C_j\}$ (note that for each clause C_j we add only one such edge), and for every $j \in \{1, \ldots, |C_i^-| - 1\}$ we add to T the edges $\{x_{ij}^{r-}, x_{i(j+1)}^{r-}\}$.

Conversely, suppose now that H contains an arborescence T rooted at r with cost at most 0 and let us define a satisfying assignment of ϕ. Since for every $i, j \in \{4, 5, 6\}$ with $i \neq j$ we have that $cc(i, j) = 1$, for every clause-vertex C_j exactly one of its incident edges belongs to T. Since $cc(i, 8) = 0$ and $cc(i, 7) = 1$ for all $i \in \{4, 5, 6\}$, we have that if $\{x_{ij}^+, C_j\}$ belongs to T, then $\{x_{ij}^+, x_{ij}^{r+}\}$, $\{x_{ij}^{r+}, x_i^r\}$, and all other edges with color 8 in the gadget corresponding to C_i^+ belong to T. Due to the same reasons, if $\{x_{ij}^-, C_j\}$ belongs to T, then $\{x_{ij}^-, x_{ij}^{r-}\}$, $\left\{x_{ij}^{r-}, x_i^{'}\right\}$, and all edges with color 8 in the gadget corresponding to C_i^- belong to T. From the structure of H and from the fact that $cc(1, 2) = 1$, it follows that in order for the tree T to span all vertices of H, for every $i \in \{1, \ldots, n\}$ either the three edges $\{x_i^\ell, x_i^+\}$, $\{x_i^+, x_i^-\}$, $\{x_i^-, x_i^r\}$ or the three edges $\{x_i^\ell, x_i^-\}$, $\{x_i^-, x_i^+\}$, $\{x_i^+, x_i^r\}$ belong to T. In the former case, we set variable x_i to 1, and in the latter case we set variable x_i to 0. Since for every $i \in \{4, 5, 6\}$ we have that $cc(7, i) = 1$, it follows that for every clause-vertex C_j, its incident edge that belongs to T joins C_j to a literal that is set to 1 by the constructed assignment. Hence, all clauses of ϕ are satisfied by this assignment, concluding the proof of the theorem.

D FPT algorithm for star tree-cut width

We have shown in Theorem 4 that the MinCCA problem is W[1]-hard on multigraphs when parameterized by tree-cut width. In this section we present an FPT algorithm for a restricted version of this parameter. Besides of being the first FPT algorithm for the problem (except for the one given in [14] parameterized by the treewidth and the maximum degree of the input graph), it provides us with the insight on the source of the problem's hardness.

Before stating the main result of this section, we introduce some preliminaries and notation used in this section. Given a graph G, a subset U of its vertices, and a subset F of its edges we denote by $G[U \cup F]$ the graph induced by these vertices and edges, that is obtained by adding to $G[U]$ the edges of F and their endpoints. Formally, $G[U \cup F] := (U \cup V(F), F \cup E(G) \cap U \times U)$.

Definition 1. *For a node t of a given tree-cut decomposition of a graph G, and two distinct vertices u, v of X_t, let $B_t^{u,v}$ be the set of children $t' \in B_t$ such that $N(Y_{t'}) = \{u, v\}$. $\hat{G}_t$ is the multigraph obtained by augmenting $G[X_t]$ with a* red *edge uv for every $t' \in B_t^{u,v}$. A* star *tree-cut decomposition is is a tree-decomposition where $\hat{G}_t$ is the disjoint union of stars, for every node t of the decomposition. The* star tree-cut width *of G is the smallest number $k \geq \mathbf{tcw}(G)$ such that every tree-cut decomposition of width at most k is a star tree-cut decomposition whenever such a k exists and n otherwise.*

We would like to stress that the above definition is somehow artificial, and we do not aim at claiming any practical application of it. The objective of the next theorem is to show that bounded tree-cut width is "almost" enough to provide an FPT algorithm for MinCCA, in the sense that if one imposes just a limited dependency among thin nodes (this is exactly what the 'star' condition is about), then an FPT algorithm is possible. It may look clear to the reader that we were looking for an FPT algorithm with parameter tree-cut width until we were able to prove Theorem 4; nevertheless, we still think that the ideas behind the proof of Theorem 7 are interesting.

Theorem 7. *The* MinCCA *problem is* FPT *parameterized by the star tree-cut width of the input graph.*

Proof: Let (G, X, χ, r, f) be an instance of MinCCA with $|V(G)| = n$ and such that $\mathbf{tcw}(G) \leq k$. We first compute, using Theorem 1, a tree-cut decomposition G of width at most $2k$, in time $2^{O(k^2 \log k)} \cdot n^2$. Then, using Theorem 2 we transform it to a *nice* tree-cut decomposition $(T, \mathcal{X})$ of width at most $2k$, in time $O(n^3)$. In order to simplify the formulation, we add to G a new vertex r' and an edge $\{r, r'\}$. We also add a node $t_{r'}$ adjacent to the unique node t_r of T whose bag contains r. We root T at this new node $t_{r'}$. Observe that the width of $(T, \mathcal{X})$ is not affected by this transformation. We color the edge $\{r, r'\}$ with a new color $\hat{x}$ that does not incur any traversal cost, i.e. $f(x, \hat{x}) = f(\hat{x}, x) = 0$ for every $x \in X$. Then, there is a one to one correspondence with the solutions of the original instance and the solutions of the modified instance. Moreover, the corresponding solutions have the same cost.

Following the general approach introduced in [11], we aim to define a data structure $\mathcal{D}_t$ at each node t of T. The dynamic programming algorithm will perform a bottom-up traversal of T, and at each node t of T different than $t_{r'}$ it will compute the table $\mathcal{D}_t$. Finally, we will extract an optimal solution from the table $\mathcal{D}_{t_r}$. Since our goal is to find a spanning tree of G, we have to analyze how such a spanning tree is decomposed by the tree-cut decomposition $(T, \mathcal{X})$.

A *partial orientation* of a set of edges is an orientation of a subset of it. Given a partial orientation Φ of $\mathsf{cut}(t)$, we denote by Φ^+ (resp., Φ^-) the set of edges in $\mathsf{cut}(t)$ that are oriented toward (resp., outward) Y_t. In our algorithm, the set Φ will correspond to the set of edges in $\mathsf{cut}(t)$ that belong to the solution. We denote by $V(\Phi^+)$ (resp. $V(\Phi^-)$) the endpoints of the edges of Φ^+ (resp.,Φ^-) in Y_t.

Let F_t be a rooted forest of $G[Y_t \cup \mathsf{cut}(t)]$ spanning all vertices in Y_t. The set of arcs $E(F_t) \cap \mathsf{cut}(t)$ is a partial orientation Φ of $\mathsf{cut}(t)$. We say that F_t *induces* Φ on $\mathsf{cut}(t)$, or simply that F_t induces Φ whenever $\mathsf{cut}(t)$ is clear from the context. We denote this by $F_t \Rightarrow \Phi$. F_t is a *good* forest if it has no sinks in Y_t. We note that a solution of the MinCCA problem, being a spanning tree rooted at r', induces a good forest on Y_t.

Let t be a node of T different than $t_{r'}$, F_t a good forest that induces some partial orientation Φ_t on $\mathsf{cut}(t)$. Since F_t does not have sinks in Y_t, every vertex of Y_t has a directed path to a sink of F_t. For every vertex $v \in X_t$ the path from v to its associated sink in F_t contains at least one edge in Φ_t^-. We now define a function $\varphi_t : \Phi_t^+ \to \Phi_t^-$ that we will term as the *partition* of Φ_t *induced* by F_t. For every edge $e \in \Phi_t^+$, $\varphi_t(e)$ is the first edge of Φ_t^- on the path from e to r'. We also say that F_t *induces* the pair (Φ_t, φ_t). Clearly, a partition of Φ_t is equivalent to a forest on $V(\Phi_t^+) \cup V(\Phi_t^-)$ consisting of stars where the vertices of $V(\Phi_t^-)$ are the centers of the stars and the vertices of $V(\Phi_t^+)$ are the leaves.

At this point now ready to define the tables that we will use in our dynamic programming algorithm. At each node t of $(T, \mathcal{X})$, our table $\mathcal{D}_t$ will store, for every partial orientation Φ_t of $\mathsf{cut}(t)$ and every function $\varphi_t : \Phi_t^+ \to \Phi_t^-$, the minimum cost of a forest inducing (Φ_t, φ_t), which we denote by $c(\Phi_t, \varphi_t)$. Note that $|\mathcal{D}_t| \leq 3^{2k} \cdot (2k)^{2k} = 2^{O(k \log k)}$. We note that $\mathsf{cut}(t_r) = \{\{r, r'\}\}$, thus every spanning tree rooted at r' induces the partial orientation Φ_r that orients $\{r, r'\}$ from r to r', i.e. $\Phi_r^- = \{\{r, r'\}\}$. Since $\Phi_r^+ = \emptyset$, then every spanning tree induces the empty partition. Therefore, the optimum of the instance is equal to the value $c(\Phi_r, \emptyset)$ in $\mathcal{D}_{t_r}$.

We now proceed to describe how the values $c(\Phi, \varphi)$ can be recursively computed in a bottom-up way. Assume first that t is a leaf of T, and fix a partial orientation Φ_t of $\mathsf{cut}(t)$. Let F_t be a good forest of t that induces Φ_t. The forest F_t can be constructed

by an appropriate choice of a parent from $X_t \setminus V(\Phi_t^+)$ to each vertex of $X_t \setminus V(\Phi_t^-)$. We iterate over all such choices (at most $(2k)^{2k}$ in number), and for each choice we a) verify in $O(k)$ time, that the choice F_t is indeed a forest, by comparing the number of connected components of F_t to $\left|\Phi_t^-\right|$, b) compute in $O(k)$ time the cost $rc(F_t)$ of F_t and finally set $c(\Phi_t, \varphi_t) = \min\{c(\Phi_t, \varphi_t), rc(F_t)\}$ in $\mathcal{D}_t$ where φ_t is the partition induced by F_t. That is,

$$c(\Phi_t, \varphi_t) = \min\{rc(F_t) \mid F_t \Rightarrow (\Phi_t, \varphi_t)\}.$$

The crucial part of the algorithm is how to perform the inductive step. That is, assuming that the tables for all the children of a node t of $(T, \mathcal{X})$ have been already computed, we have to show that the table at node t can be computed in FPT-time. Consult Fig. 11 for the following discussion. Let t be a non-leaf node of T, and F_t be a forest of $G[Y_t \cup \mathsf{cut}(t)]$. F_t induces

- a partial orientation Φ_t of $\mathsf{cut}(t)$,
- a partial orientation Φ of $\delta(X_t)$,
- a partition $\varphi_t : \Phi_t^+ \to \Phi_t^-$,
- a forest F of $G[X_t \cup \delta(X_t)]$, and
- for every child node t' of t
  - a forest $F_{t'}$ of $G[Y_{t'} \cup \mathsf{cut}(t')]$,
  - a partial orientation $\Phi_{t'}$ of $\mathsf{cut}(t')$,
  - a partition $\varphi_{t'} : \Phi_{t'}^+ \to \Phi_{t'}^-$.

Therefore, we proceed as follows:

- Guess a partial orientation Φ_t among the $3^{2k} = 2^{O(k)}$ partial orientations of $\mathsf{cut}(t)$.
- Guess a partial orientation $\hat{\Phi}$ of $\cup_{t' \in A_t} \mathsf{cut}(t') \setminus \mathsf{cut}(t)$. Note that, since $|A_k| \leq 4k+1$ by Equation (1), the number of possibilities is at most $3^{(4k+1)2k} = 2^{O(k \log k)}$.
- $\Phi_t \cup \hat{\Phi}$ induces a partial orientation Φ of $\delta(X_t)$ and partial a orientation $\Phi_{t'}$ for every $t' \in A_t$.
- For every $t' \in A_t$ guess a function $\varphi_{t'} : \Phi_{t'}^+ \to \Phi_{t'}^-$. Note that the number of possible guesses is at most $((2k)^{2k})^{4k+1} = 2^{O(k^2 \log k)}$.
- Guess a forest F of $G[X_t, \delta(X_t)]$ that induces Φ by choosing a parent for each vertex of $X_t \setminus V(\Phi^-)$ and counting the number of connected components. The number of possible guesses is at most $(2k)^{2k} = 2^{O(k \log k)}$.
- Verify that $\hat{F} = F \cup (\cup_{t' \in A_t}(\Phi_{t'} \cup \varphi_{t'}))$ is a good forest.
- Let φ_t be the partition that $\hat{F}$ induces on Φ_t.

First, we assume for simplicity that $B_t = \emptyset$. It is important to note that F_t induces both F and (Φ_t, φ_t) if and only if $F_t = F \cup (\cup_{t' \in A_t} F_{t'})$ where each forest $F_{t'}$ induces $(\Phi_{t'}, \phi_{t'})$ on $\mathsf{cut}(t')$. Moreover, $rc(F_t) = rc(F) + \sum_{t' \in A_t} rc(F_{t'})$. Therefore, the minimum cost forest F_t that can be obtained by this set of guesses is

$$\hat{c} = rc(F) + \sum_{t' \in A_t} c(\Phi_{t'}, \varphi_{t'}). \tag{2}$$

Finally, we set $c(\Phi_t, \varphi_t) = \min\{c(\Phi_t, \varphi_t), \hat{c}\}$ in the table $\mathcal{D}_t$.

To this end, our algorithm is to iterate over all guesses above, for each valid guess to compute its cost $\hat{c}$ and to store for each pair (Φ_t, φ_t) the smallest value $\hat{c}$ associated with this pair. We note that the number of guesses is at most $2^{O(k^2 \log k)}$, and that the computations involved in each guess can be performed in polynomial time.

It remains to deal with the children of t in the set B_t, which can be arbitrarily many (that is, not necessarily bounded by a function of k). Recall that for every such child $t' \in B_t$, it holds that $\mathsf{adh}(t') \leq 2$ and that $N(Y_{t'}) \subseteq X_t$. Since we can assume that G is connected, we have that $\mathsf{adh}(t') \geq 1$ for every child $t' \in B_t$. Let $t' \in B_t$ such that $\mathsf{adh}(t') = 1$, and let $\mathsf{cut}(t') = \{e_{t'}\}$. Clearly, the edge $e_{t'}$ belongs to any arborescence spanning G, in particular to F_t. In other words the only partial orientation to be considered for $\mathsf{cut}(t')$ is $\Phi^-_{t'} = \{e_{t'}\}$ and the only possible partition is $\emptyset$. It is sufficient to add the term $c(\Phi^-_{t'}, \emptyset)$ to the right hand side of Equation (2).

We now consider $t' \in B_t$ with $\mathsf{adh}(t') = 2$. Assume that t' is such that both edges in $\mathsf{cut}(t')$, say e_1 and e_2, are incident to the same vertex v in X_t. Clearly, none of e_1 and e_2 can be oriented towards X_t, and therefore there are only three potential partial orientations of $\mathsf{cut}(t')$ that can possibly be induced by F_t are $\Phi^-_{t',1} = \{e_1\}$, $\Phi^-_{t',2} = \{e_2\}$, or $\Phi^-_{t',3} = \{e_1, e_2\}$, and $\Phi^+_{t',1} = \Phi^+_{t',2} = \Phi^+_{t',3} = \emptyset$. Thus $\varphi_{t',1} = \varphi_{t',2} = \varphi_{t',3} = \emptyset$. Let e_v be the edge leading v to its parent in $\hat{F}$. Then, for $i \in [3]$ the i-th possibility contributes the value $c(\Phi_{t',i}, \emptyset) + \sum_{e_i \in \Phi^-_{t',i}} f(e_v, e_i)$. Therefore, we add to the right hand side of Equation (2) the term

$$\min\{c(\Phi_{t',i}, \emptyset) + \sum_{e_i \in \Phi^-_{t',i}} f(e_v, e_i) |\ 1 \leq i \leq 3\}.$$

To this end our algorithm remains intact, except the computation of $\hat{c}$ for each guess that now contains additional terms. We remain with the case that the two edges of $\mathsf{cut}(t')$ are incident to two distinct vertices u, v of X_t. Recall that in this case $t' \in B_t^{u,v}$ (the node t'_5 in Fig. 11 is an example of such a node).

Let $\mathsf{cut}(t') = \{e_1, e_2\}$ where $e_1 = \{u, u'\}$ and $e_2 = \{v, v'\}$ (note that possibly $u' = v'$). In this case there are five potential partial orientations of $\mathsf{cut}(t')$ that F_t can possibly induce, namely the partial orientations $\Phi_{t',1}, \Phi_{t',2}, \Phi_{t',3}$ as in the previous case, and two partial orientations $\Phi_{t',uv}$ and $\Phi_{t',vu}$ that orient one edge towards Y_t and the other towards X_t. For each one of the last two cases there is exactly one possible partition: $\varphi_{t',uv}$ and $\varphi_{t',vu}$, respectively. We consider two cases.

- $F_{t'}$ induces $(\Phi_{t',i}, \varphi_{t',i})$ for some $i \in [3]$: In this case we can choose the best value for i as we did before.
- $F_{t'}$ induces $(\Phi_{t',uv}, \varphi_{t',uv})$ where $\Phi_{t',uv}$ orients e_1 from u to u' and e_2 from v' to v: We will modify F_t to obtain another "equivalent" forest which does not span $Y_{t'}$ but contains edges that not present in G. We remove $F_{t'}$ from F_t and replace it by a *simulating* arc $\hat{e}_{t',uv}$ from u to v. Furthermore, we assign to this arc a weight $w(\hat{e}_{t',uv}) = c_{t'}(\Phi_{t',uv}, \varphi_{t',uv})$ that corresponds to the cost of traversing the entire tree $F_{t'}$ (starting from its root u and ending at the leaves including v), i.e., $c(\Phi_{t',uv}, \varphi_{t',uv})$. In order to simulate the two traversal costs at the vertices u and v, we assign to $\hat{e}_{t',uv}$ a unique color $x_{t',uv}$ such that the cost of entering (resp., leaving) this edge is equal to the cost of entering e_1 (resp., e_2). That is, $f(x, x_{t',uv}) = f(x, \chi(e_1))$ and $f(x_{t',uv}, x) = f(\chi(e_2), x)$ for every color $x \in X$. Let $\hat{F}_t$ be the forest obtained by repeating this transformation for every pair u, v and every $F_{t'}$ that induces one of $(\Phi_{t',uv}, \varphi_{t',uv})$, $(\Phi_{t',vu}, \varphi_{t',vu})$. By the construction, we have that $rc(F_t) = rc(\hat{F}_t) + w(\hat{F}_t)$. Note that the simulating arcs correspond to the red edges of $\hat{G}_t$.

Therefore, it would be sufficient to modify the way F is guessed so that it allows F to contain red edges of $\hat{G}_t$. However, since the number of these edges is not bounded by a function of k, this would not imply an FPT.

To cure this problem, we define $\hat{\hat{G}}_t$ as the multi-graph obtained from $\hat{G}_t$ by replacing every set of parallel red edges by one red edge. We also allow F to contain a red edge of $\hat{\hat{G}}_t$. Since the number of edges of $\hat{\hat{G}}_t$ is bounded by a function of k, the number of choices for F remains a function of k. In the sequel, our goal is to find the best forest F_t that induces F and all the guessed values. In other words we want to find a) a red edge of $\hat{G}_t$ for each red edge of F, and b) for every other red edge of $\hat{G}_t$ to decide upon one of the possible partial orientations.

In the sequel we show how to make these decisions (namely a) and b) above) using a dynamic programming algorithm that performs a bottom-up traversal of F. We first make the following simplifying assumption: every red edge of $\hat{\hat{G}}$ is in F. For $v \in X_t$, let $F_{t,v}$ be the subtree of F_t rooted at v. Let $e = (u, v)$ be an arc of F, and $\hat{e}$ be any edge of $\hat{G}$ between u and v. We denote by $F_{t,\hat{e}}$ the tree consisting of $F_{t,v}$ and the arc $\hat{e}$. We discard costs incurred by traversals in vertices in subtrees $Y_{t'}$ since they are fixed by the current set of guesses. We compute in a bottom-up fashion the values $OPT_F(\hat{e})$ that denote the minimum cost of the subtree $F_{t,\hat{e}}$ among all trees F_t that agree with the current set of guesses. Note that for a node $t' \in B_t^{u,v}$, once the inbound edges of u and v are fixed, we can decide on the best partial orientation $\Phi_{t'}$ by comparing the three possible values.

If v is a leaf of F then $F_{t,\hat{e}}$ contains at most on type a) red edge, namely $\hat{e}$, and no other red edges, by our assumption. Therefore, $OPT_F(\hat{e})$ can be computed by summing up the traversal costs between $\hat{e}$ and all the edges of $\hat{\Phi}$ oriented from v outside. If v is not a leaf of F, let $e_1, \ldots, e_\ell$ be the arcs from v to its children. For each such arc e_j we have to choose exactly one arc $\hat{e_j}$ of $\hat{G}$. The crucial point is that these choices can be made independently of each other. For every possible choice for $\hat{e_j}$ we compute the cost incurred by traversing v from $\hat{e}$ to $\hat{e_j}$ and for each arc $\hat{e_{j'}} \neq \hat{e_j}$ we add the minimum among the three possible costs. Note that these costs can be computed since the two edges incident to $\hat{e_{j'}}$ are known (namely $\hat{e}$ and $\hat{e_j}$) at this point. We choose the arc $\hat{e_j}$ that leads to the smallest cost, and we repeat this for every edge e_j and sum up the costs.

Finally, it remains to relax the simplifying assumption, i.e. to handle the edges of $\hat{\hat{G}}$ that are not in F. Let $\hat{e} = u$ be an edge of $\hat{G}$ that corresponds to such an edge. There are only three possible ways to traverse the subtree corresponding to $\hat{e}$, in order to choose the best one we have to know the edges e_u, e_v of F_t leading u and v to their parents, respectively. Since the red edges of $\hat{G}$ constitute a union of stars, at least one of e_u, e_v is not red, therefore its color is known. Then, the cost of $\hat{e}$ depends only on the choice of one (red) edge. We can modify the above dynamic programming algorithm so that the cost of each such edge is associated with the red edge in F incident to it. If no such edge exists, then the cost of $\hat{e}$ is constant for the current choice of F. □

E Figures

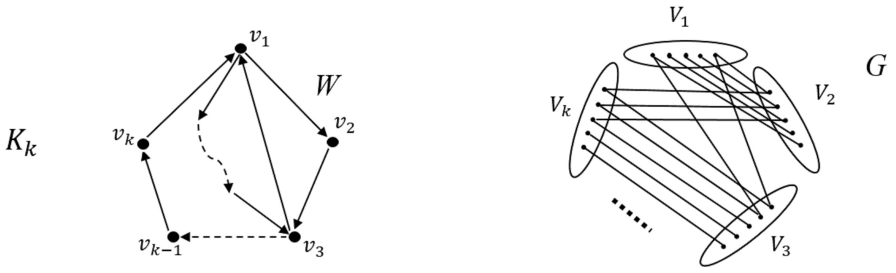

Fig. 2. Construction in the proof of Theorem 3: The complete graph K_k and an Eulerian circuit W in K_k starting with $v_1, v_2, \ldots, v_k, v_1$ and ending with v_3, v_1. A k-colored graph G is also illustrated.

Fig. 3. Construction in the proof of Theorem 3: A left-to-right representation of the Eulerian circuit W.

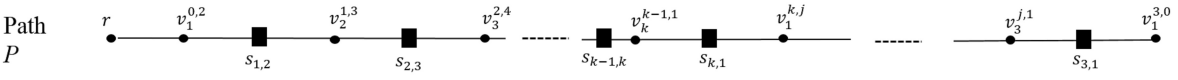

Fig. 4. Construction in the proof of Theorem 3: Path P on $2\binom{k}{2}+2$ vertices. Selection vertices are depicted by squares.

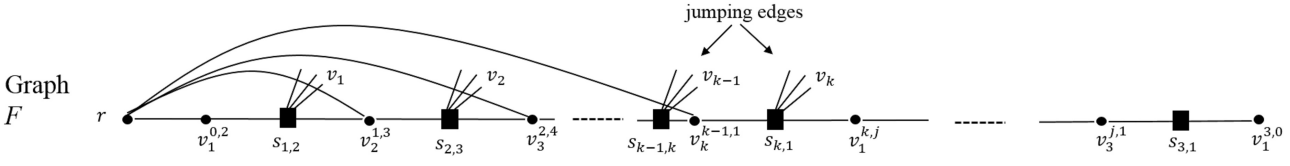

Fig. 5. Construction in the proof of Theorem 3: Graph F.

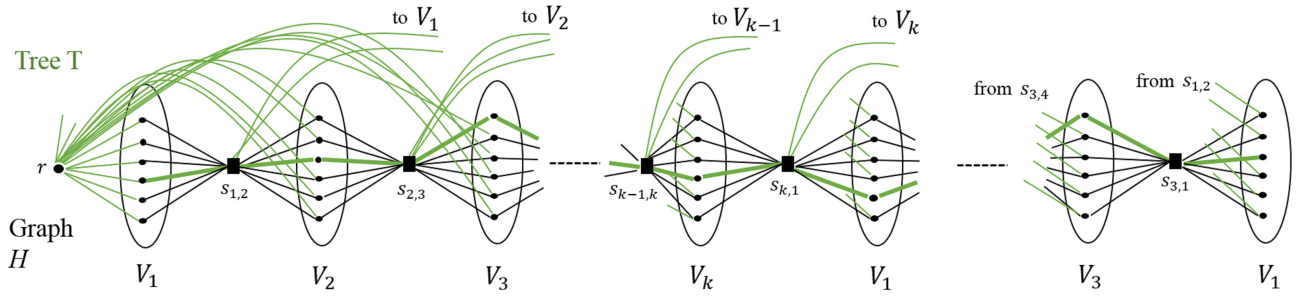

Fig. 6. Construction in the proof of Theorem 3: Graph H and a solution arborescence T drawn in green.

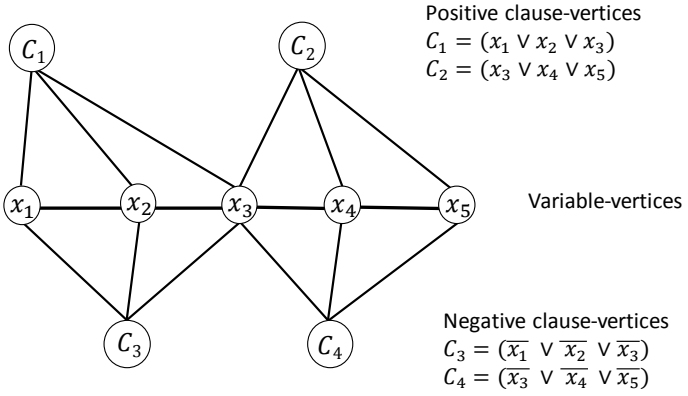

Fig. 7. A planar monotone rectilinear 3-SAT instance.

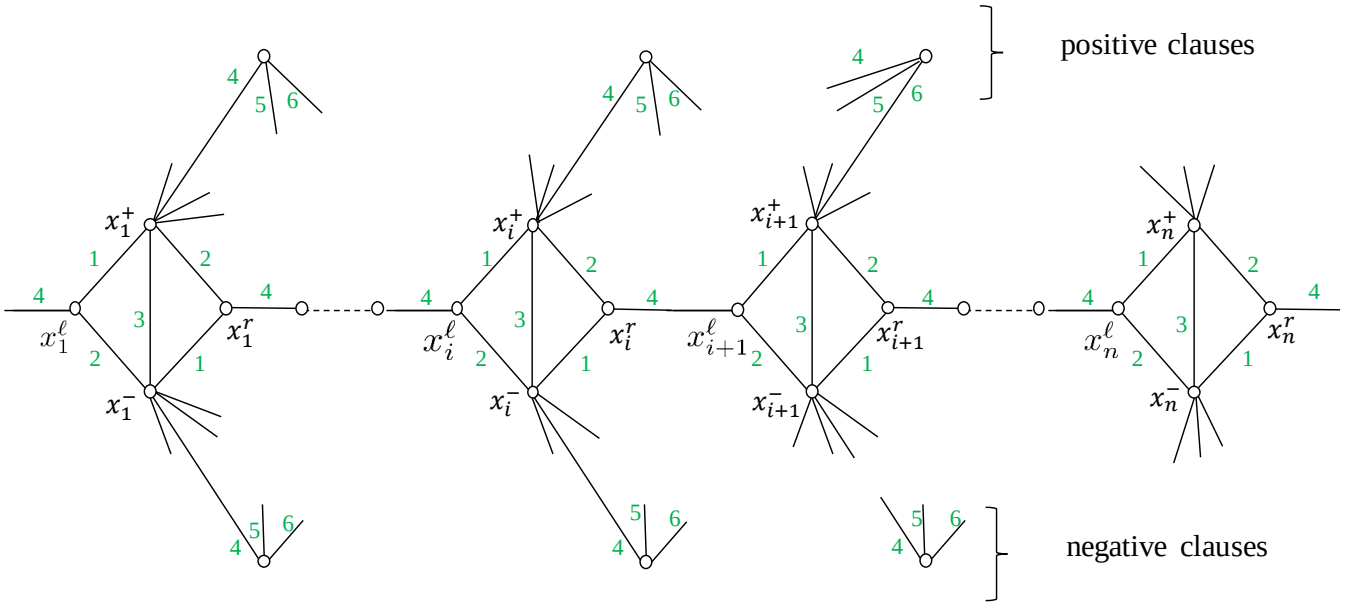

Fig. 8. The graph H constructed in the proof of Theorem 5, together with the edge-coloring function χ (in green).

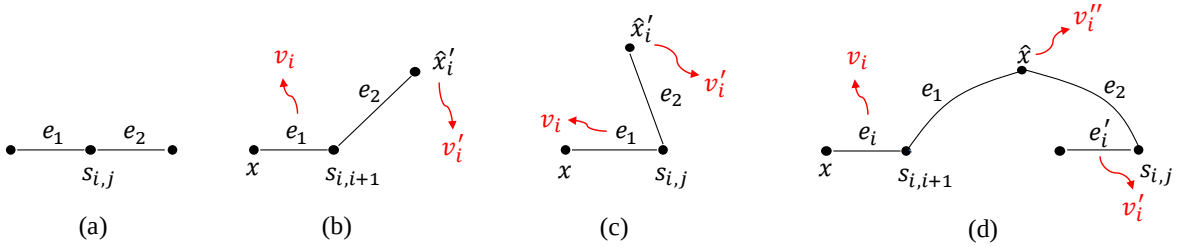

Fig. 9. (a)-(b)-(c) The three cases where $cc(e_1, e_2) = 1$ in the proof of Theorem 4. (d) Construction in the proof. The red vertices in the figure correspond to the vertices in V_i that are associated with the corresponding edges or vertices of H.

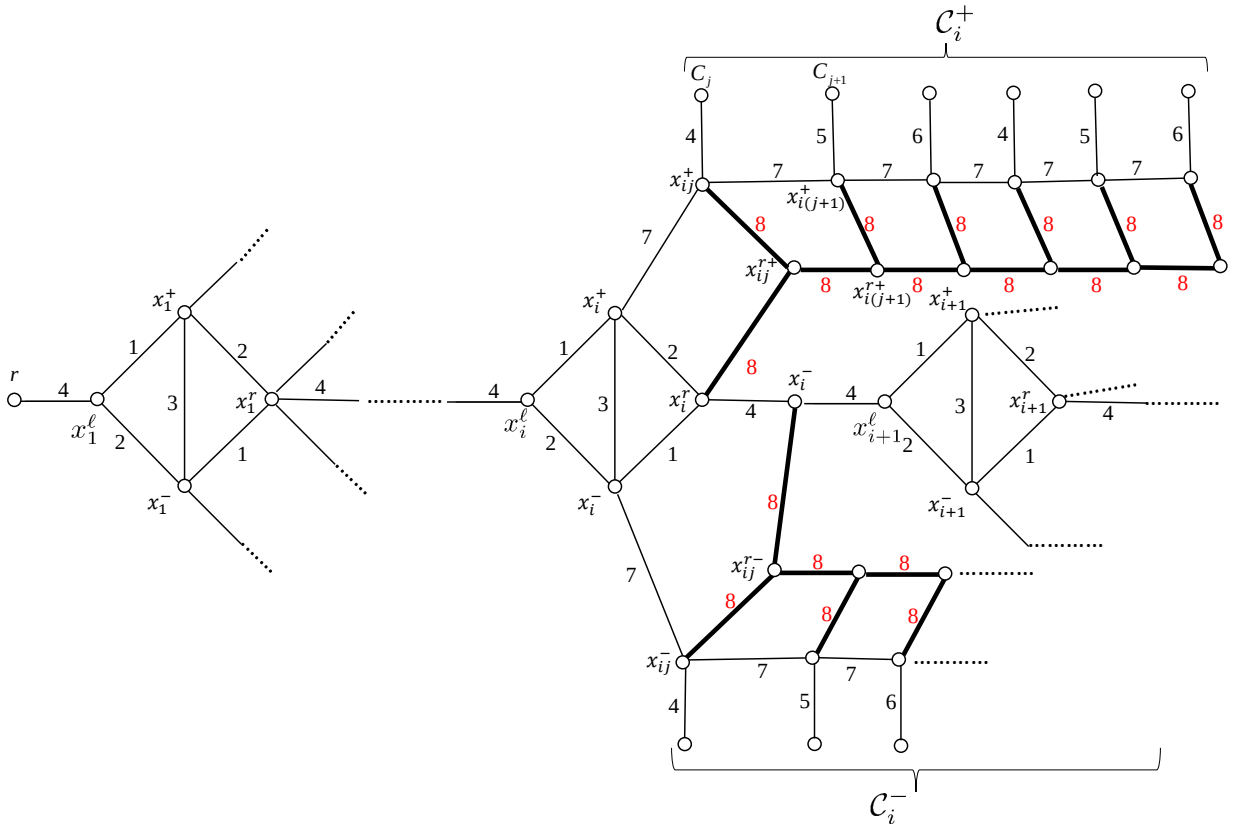

Fig. 10. The graph H constructed in the proof of Theorem 6, together with the edge-coloring function χ. Note that the maximum degree of H is 4.

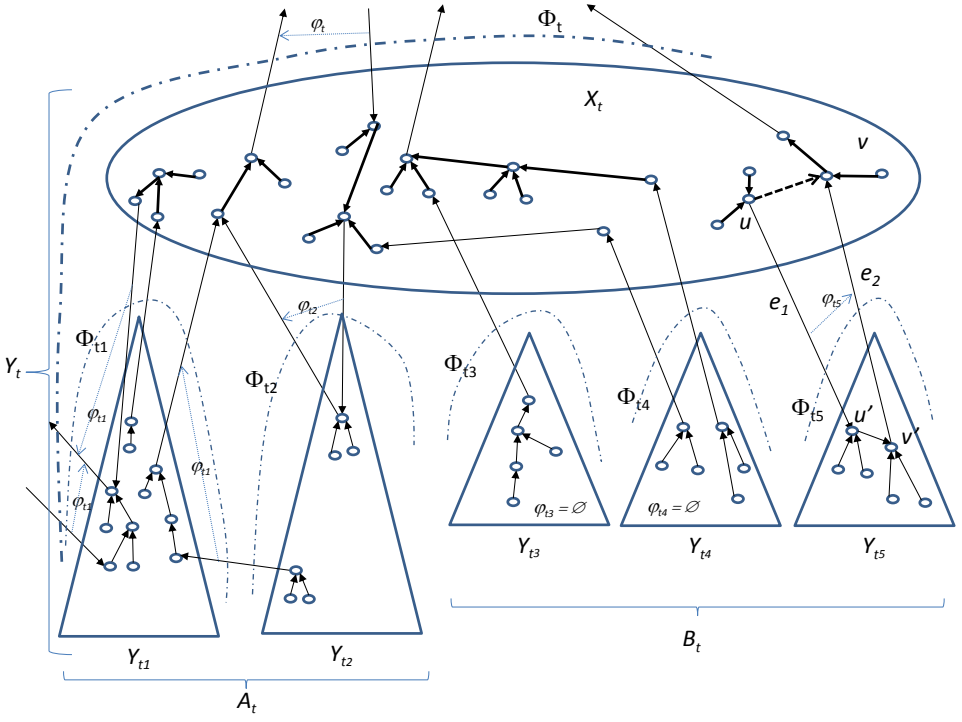

Fig. 11. The decomposition of a forest F_t associated with a node t of a tree-cut decomposition in the proof of Theorem 7.