The Parameterized Complexity of the Minimum Shared Edges Problem*

Till Fluschnik[1], Stefan Kratsch[2], Rolf Niedermeier[1], and Manuel Sorge[1]

1 **Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany,** {till.fluschnik, rolf.niedermeier, manuel.sorge}@tu-berlin.de
2 **Institut für Informatik, Universität Bonn, Germany,** kratsch@cs.uni-bonn.de

Abstract

We study the NP-complete Minimum Shared Edges (MSE) problem. Given an undirected graph, a source and a sink vertex, and two integers p and k, the question is whether there are p paths in the graph connecting the source with the sink and sharing at most k edges. Herein, an edge is shared if it appears in at least two paths. We show that MSE is W[1]-hard when parameterized by the treewidth of the input graph and the number k of shared edges combined. We show that MSE is fixed-parameter tractable with respect to p, but does not admit a polynomial-size kernel (unless NP $\subseteq$ coNP/poly). In the proof of the fixed-parameter tractability of MSE parameterized by p, we employ the treewidth reduction technique due to Marx, O'Sullivan, and Razgon [ACM TALG 2013].

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory

Keywords and phrases Parameterized complexity, kernelization, treewidth, treewidth reduction

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2015.448

1 Introduction

We consider the parameterized complexity of the following basic routing problem.

Minimum Shared Edges (MSE)
Input: An undirected graph $G = (V, E)$, $s, t \in V$, $p \in \mathbb{N}$ and $k \in \mathbb{N}_0$.
Question: Is there a (p, s, t)-routing in G in which at most k edges are shared?

Herein, a (p, s, t)-*routing* is a cardinality-p set of s-t paths, and an edge is called *shared* if it is contained in at least two of the paths in the routing. If s and t are understood from the context, we simplify notation and speak of p-routings and refer to the paths it contains as *routes*. Minimum Shared Edges is polynomial-time solvable with $k = 0$, while it becomes NP-hard for general values of k [14].

Minimum Shared Edges has two natural applications. One is to route an important person which is under threat of attack from s to t in a street network. In order to confound attackers, $p - 1$ additional, empty convoys are routed, and guards are placed on streets that are shared by routes. Minimum Shared Edges then minimizes the costs to place guards [20]. A second application arises from finding a resilient way of communication

* Due to space constraints numerous details are deferred to a full version. Till Fluschnik, Stefan Kratsch, and Manuel Sorge gratefully acknowledge support by the DFG, projects DAMM (NI 369/13-2), PREMOD (KR 4286/2-1), and DAPA (NI 369/12-2), respectively.

between two servers s and t in an interconnection network, assuming that $p-1$ faulty connections may be present that block or alter the communicated information. Finding p edge-disjoint paths ensures at least one piece of information arrives unscathed. When this is not possible, and if we can ensure that a link is not faulty by expending some fixed cost per link, then Minimum Shared Edges is the problem of finding a resilient way of communication that minimizes the overall costs [21].

We study Minimum Shared Edges from a parameterized complexity perspective, that is, for certain parameters ℓ of the inputs (of size r), we identify algorithms with running time $f(\ell) \cdot \text{poly}(r)$ or we prove that such algorithms are unlikely to exist. There are two natural parameters for Minimum Shared Edges: the number p of routes and the number k of shared edges. Both of them can be reasonably assumed to be small in applications. As we will see, there is also a connection between p and the treewidth tw of G.

Related Work. Omran et al. [20] introduced Minimum Shared Edges on directed graphs and showed NP-hardness by a reduction from Set Cover. The reduction also implies W[1]-hardness with respect to the number k of shared arcs in this directed case. Undirected Minimum Shared Edges admits an XP-algorithm with respect to treewidth, more specifically, it can be solved in $O((n+m) \cdot (p+1)^{2^{\omega \cdot (\omega+1)/2}})$ time [24], where ω upper-bounds the treewidth of the input graph.

Assadi et al. [2] introduced a generalization of directed Minimum Shared Edges, called Minimum Vulnerability, which additionally considers arc weights (the cost of sharing an arc), arc capacities (an upper bound on the number of routes supported by an arc) and a share-threshold for each arc (the threshold of routes, possibly other than two, after which the arc becomes shared). Directed Minimum Vulnerability admits an XP-algorithm with respect to the number p of routes [2]. Undirected Minimum Vulnerability is NP-hard even on bipartite series-parallel graphs, but admits a pseudo-polynomial-time algorithm on bounded treewidth graphs [1]. Furthermore, Minimum Vulnerability is fixed-parameter tractable with respect to p on chordal graphs [1].

There are also several results regarding approximation algorithms and lower bounds [2, 20]; however, our focus is on exact algorithms.

Our Contributions. We present two main results: Minimum Shared Edges is fixed-parameter tractable (FPT) with respect to the number p of routes and it is W[1]-hard with respect to the treewidth tw and the number k of shared edges combined. Moreover, complementing the fixed-parameter tractability result with respect to p, we show that there is no polynomial-size problem kernel with respect to p (Section 5), unless NP $\subseteq$ coNP/poly.

The FPT result with respect to p is obtained by modifying the input graph so that the resulting graph has treewidth bounded by some (exponential) function of p using the treewidth reduction technique [17] (see Section 4). Then we apply a dynamic program which also is an FPT algorithm with respect to p and tw (Section 3). For this purpose, we design a new dynamic program rather than using the ones from the literature [1, 2]. In comparison, ours yields an improved running time in the FPT algorithm with respect to p, that is, the dependence is doubly exponential on p rather than triply exponential. Our result complements the known FPT algorithm for undirected Minimum Vulnerability on chordal graphs, parameterized by p [1]. Treewidth reduction has lately also found applications in a wide variety of other problems, for example, in graph coloring [5], graph partitioning [3], and arc routing [15].

As mentioned, our second main result is that Minimum Shared Edges is W[1]-hard with respect to the treewidth tw and the number k of shared edges combined. This provides a

corresponding lower bound for the known polynomial-time algorithms on constant-treewidth graphs for Minimum Shared Edges and for the more general undirected Minimum Vulnerability [1, 2]. More precisely, the degree of the polynomial in the running time depend on tw and our result shows that removing this dependence is impossible unless FPT = W[1]. Interestingly, the known dynamic programs on tree decompositions keep track of the number of routes over certain separators in their tables. Our hardness result shows that information of this sort is crucial.

2 Preliminaries

We use basic notation from parameterized complexity [6, 10, 13, 18] and graph theory [8, 23].

Graphs and Tree Decompositions. Unless stated otherwise, all graphs are without parallel edges or loops. When it is not ambiguous, we use n for the number of vertices of a graph and m for the number of edges.

Let $G = (V, E)$ be an undirected graph. We write $V(G)$ for the vertex set of graph G and $E(G)$ for the edge set of graph G. We define the size of graph G as $|G| := |V(G)|+|E(G)|$. For a vertex set $W \subseteq V(G)$, we denote by $G[W]$ the subgraph of G *induced* by the vertex set W. For an edge set $F \subseteq E(G)$, we denote by $G[F]$ the subgraph of G *induced* by the edge set F. We write G/F and $G\backslash F$ for the contraction and the deletion of the edges in F, respectively (we write G/e and $G\backslash e$ for short if $F = \{e\}$).

A *tree decomposition* of a graph G is a tuple $\mathbb{T} := (T, (B_\alpha)_{\alpha \in V(T)})$ of a tree T and family $(B_\alpha)_{\alpha \in V(T)}$ of sets $B_\alpha \subseteq V(G)$, called *bags*, such that

(i) $V(G) = \bigcup_{\alpha \in V(T)} B_\alpha$,
(ii) for every edge $e \in E(G)$ there exists an $\alpha \in V(T)$ such that $e \subseteq B_\alpha$ and
(iii) for each $v \in V(G)$, the graph induced by the node set $\{\alpha \in V(T) \mid v \in B_\alpha\}$ is a tree.

The *width* ω of a tree decomposition $\mathbb{T}$ of a graph G is defined as $\omega(\mathbb{T}) := \max\{|B_\alpha|-1 \mid \alpha \in V(T)\}$. The *treewidth* $\mathsf{tw}(G)$ of a graph G is the minimum width over all tree decompositions of G. A tree decomposition $\mathbb{T} = (T, (B_\alpha)_{\alpha \in V(T)})$ is a *nice tree decomposition with introduce edge nodes* if the following conditions hold.

(i) The tree T is rooted and binary.
(ii) For each edge in $E(G)$ there is exactly one *introduce edge node* in $\mathbb{T}$, where an introduce edge node is a node α in the tree decomposition $\mathbb{T}$ of G labeled with an edge $\{v, w\} \in E(G)$ with $v, w \in B_\alpha$ that has exactly one child node α'; furthermore $B_\alpha = B_{\alpha'}$.
(iii) Each node $\alpha \in V(T)$ is of one of the following types:
- introduce edge node;
- *leaf node*: α is a leaf of T and $B_\alpha = \emptyset$;
- *introduce vertex node*: α is an inner node of T with exactly one child node $\beta \in V(T)$; furthermore $B_\beta \subseteq B_\alpha$ and $|B_\alpha\backslash B_\beta| = 1$;
- *forget node*: α is an inner node of T with exactly one child node $\beta \in V(T)$; furthermore $B_\alpha \subseteq B_\beta$ and $|B_\beta\backslash B_\alpha| = 1$;
- *join node*: α is an inner node of T with exactly two child nodes $\beta, \gamma \in V(T)$; furthermore $B_\alpha = B_\beta = B_\gamma$.

A given tree decomposition can be modified in linear time to fulfill the above constraints; moreover, the number of nodes in such a tree decomposition of width ω is $O(\omega \cdot n)$ [16, 7].

Cuts and Paths. Let G be an undirected, connected graph. A *cut* $C \subseteq E$ is a set of edges such that the graph $G\backslash C$ is not connected. Let $s, t \in V(G)$ be two vertices in G. An *s-t cut* C

is a cut such that the vertices s and t are not connected in $G \backslash C$. A *minimum s-t cut* is an s-t cut C such that $|C| = \min |C'|$, where the minimum is taken over all s-t cuts C' in G. An s-t cut C in G is *minimal* if for all edges $e \in C$ it holds that $C \backslash \{e\}$ is not an s-t cut in G.

A *path* is a connected graph with exactly two vertices of degree one and no vertex of degree at least three. We call the vertices with degree one the *endpoints* of the path. The *length* of a path is defined as the number of edges in the path. For two distinct vertices $s, t \in V(G)$, we refer to the path with endpoints s and t (as subgraph of G) as s-t path in G.

Parameterized Complexity. A *parameterized problem* is a set of instances $(\mathcal{I}, \ell)$, where $\mathcal{I} \in \Sigma^*$ for a finite alphabet Σ, and $\ell \in \mathbb{N}$ is the *parameter*. A parameterized problem Q is *fixed-parameter tractable*, shortly FPT, if there exists an algorithm that on input $(\mathcal{I}, \ell)$ decides whether $(\mathcal{I}, \ell)$ is a yes-instance of Q in $f(\ell) \cdot |\mathcal{I}|^{O(1)}$ time, where f is a computable function independent of $|\mathcal{I}|$.

W$[t]$, $t \geq 1$, are classes that (amongst others) contain parameterized problems which presumably do not admit FPT algorithms. Hardness for W$[t]$ can be shown by reducing from a W$[t]$-hard problem, using a *parameterized* reduction, that is, a many-to-one reduction that runs in FPT time and maps any instance $(\mathcal{I}, \ell)$ to another instance $(\mathcal{I}', \ell')$ such that $\ell' \leq f(\ell)$ for some computable function f.

A parameterized problem Q is *kernelizable* if there exists a polynomial-time self-reduction that maps an instance $(\mathcal{I}, \ell)$ of Q to another instance $(\mathcal{I}', \ell')$ of Q such that: (1) $|\mathcal{I}'| \leq \lambda(\ell)$ for some computable function λ, (2) $\ell' \leq \lambda(\ell)$, and (3) $(\mathcal{I}, \ell)$ is a yes-instance of Q if and only if $(\mathcal{I}', \ell')$ is a yes-instance of Q. The instance $(\mathcal{I}', \ell')$ is called the *problem kernel* of $(\mathcal{I}, \ell)$ and λ is called its *size*.

3 An Algorithm for Small Treewidth and Small Number of Paths

In this section we present the following theorem.

▶ **Theorem 1.** *Let G be a graph with $s, t \in V(G)$ given together with a tree decomposition of width ω. Let $p \in \mathbb{N}$ be an integer. Then the minimum number of shared edges in a (p, s, t)-routing can be computed in $O(p \cdot (\omega + 4)^{3 \cdot p \cdot (\omega+3)+4} \cdot n)$ time.*

The proof is based on a dynamic program that computes a table for each node of the (arbitrarily rooted) tree decomposition in a bottom-up fashion. For our application, it is convenient to use a nice tree decomposition with introduce edge nodes such that each bag contains the sink and the source node. For each node α in the tree decomposition $\mathbb{T}$ of G, we define V_α as the set of vertices and E_α as the set of edges that are introduced in the subtree rooted at node α. In other words, a vertex $v \in V(G)$ is in V_α if and only if there exists at least one introduce vertex node in the subtree rooted at node α that introduced vertex v. As a special case, since the vertices s and t are contained in every bag, we consider s and t as introduced by each leaf node. An edge $e \in E(G)$ is in E_α if and only if there exists an introduce edge node in the subtree rooted at node α that introduced edge e. Recall that there is a unique introduce edge node for every edge of graph G. We define $G_\alpha := (V_\alpha, E_\alpha)$ as the graph for node α. For every leaf node α in $\mathbb{T}$, we set $V_\alpha = \{s, t\}$ and $E_\alpha = \emptyset$.

Partial Solutions. We define a set of p forests in G_α as a *partial solution* L_α for node α. Instead of asking for p s-t routes that share at most k edges, we can ask for p s-t forests that share at most k edges, where an s-t forest is a forest that contains at least one tree connecting

vertices s and t. Note that every forest that contains a tree containing both vertices s and t can be "reduced" to an s-t path. A partial solution L_α has a cost value $c(L_\alpha)$, which is the number of edges in G_α that appear in at least two of the p forests in L_α.

In order to represent the intersection of the trees in a partial solution with the bag that we are currently considering, we use the following notation. For each node α in the tree decomposition $\mathbb{T}$ of G, we consider p-tuples of pairs $\mathcal{X}^\alpha := (\mathcal{Y}_q^\alpha, Z_q^\alpha)_{q=1,\ldots,p}$, where for each $q \in [p]$, $Z_q^\alpha \subseteq B_\alpha$ together with $\mathcal{Y}_q^\alpha \subseteq 2^{B_\alpha}$ is a partition of B_α, that is,

(i) $\bigcup_{M \in \mathcal{Y}_q^\alpha} M \cup Z_q^\alpha = B_\alpha$, and

(ii) for all $X, Y \in \mathcal{Y}_q^\alpha \cup \{Z_q^\alpha\}$ with $X \neq Y$ it holds that $X \cap Y = \emptyset$.

We say that $\mathcal{X}^\alpha$ is a *signature* for node α. For each $q \in [p]$, we call the pair $(\mathcal{Y}_q^\alpha, Z_q^\alpha)$ a *segmentation* of the vertex set B_α. We write segmentation q instead of segmentation with index q for short. We call each $M \in \mathcal{Y}_q^\alpha$ a *segment* of the segmentation q and we call Z_q^α the *zero*-segment of the segmentation q.

To connect signatures (and segmentations) with the partial solutions that they represent, we use the following notation. We say that the signature $\mathcal{X}^\alpha$ is a *valid* signature for node α if there is a partial solution L_α for node α such that for each $q \in [p]$, the zero-segment Z_q^α is the set of nodes in B_α that do not appear in the forest with index q and for each set $M \in \mathcal{Y}_q^\alpha$, there is a tree S in the forest with index q such that $M = B_\alpha \cap V(S)$. In other words, the sets in $\mathcal{Y}_q^\alpha$ correspond to connected components in the forest with index q of the partial solution. We say that $\mathcal{X}^\alpha$ is a signature *induced* by the partial solution L_α if $\mathcal{X}^\alpha$ is a valid signature for node α and the partial solution L_α validates $\mathcal{X}^\alpha$. In this case, for each $q \in [p]$, the pair $(\mathcal{Y}_q^\alpha, Z_q^\alpha)$ is an *induced* segmentation. We remark that given $\mathcal{X}^\alpha$, there can be exactly one, more than one, or no partial solution with signature $\mathcal{X}^\alpha$. Given a partial solution L_α for G_α, there is exactly one signature induced by L_α. Let $\mathcal{X}^\alpha$ be a signature for node α such that there is no partial solution for G_α that induces the signature $\mathcal{X}^\alpha$, then we say that $\mathcal{X}^\alpha$ is an *invalid* signature.

Let $\mathbb{T} = (T_\mathbb{T}, (B_\alpha)_{\alpha \in V(T_\mathbb{T})})$ be a nice tree decomposition of G with introduce edge nodes and vertices s and t contained in every bag. Let $\omega := \omega(\mathbb{T})$ be the width of $\mathbb{T}$. We consider the table T in the following dynamic program that we process bottom-up on the tree decomposition $\mathbb{T}$, that is, we start to fill the entries of the table T at the leaf nodes of the tree decomposition $\mathbb{T}$ and we traverse the tree of the tree decomposition from the leaves to the root. For a node α in the tree decomposition $\mathbb{T}$ and a signature $\mathcal{X}^\alpha$ for node α, the entry $T[\alpha, \mathcal{X}^\alpha]$ is defined as

$$T[\alpha, \mathcal{X}^\alpha] := \begin{cases} \min c(L_\alpha), & \text{if } \mathcal{X}^\alpha \text{ is a valid signature,} \\ \infty, & \text{otherwise,} \end{cases}$$

where the minimum is taken over all partial solutions L_α in G_α such that L_α induces the signature $\mathcal{X}^\alpha$.

For each type of node in $\mathbb{T}$, we define a rule on how to fill each entry in T, and discuss the running time for applying the rule and the running time for filling all entries in T for the given type of node. Due to space constraints, we give some details only for introduce edge nodes, and defer the correctness proof and the remaining nodes to the full version of the paper.

Introduce Edge Node. Let α be an introduce edge node of $\mathbb{T}$, let β be the child node of α, and let $e = \{v, w\}$ be the edge introduced by node α. Two signatures $\mathcal{X}^\alpha$ and $\mathcal{X}^\beta$ are *compatible* if for each $q \in [p]$, one of the following conditions holds:

(i) $\mathcal{Y}_q^\alpha = \mathcal{Y}_q^\beta$, or

(ii) $\mathcal{Y}_q^\alpha = (\mathcal{Y}_q^\beta \backslash \{M_1, M_2\}) \cup \{M_1 \cup M_2\}$ with $M_1, M_2 \in \mathcal{Y}_q^\beta$, $M_1 \neq M_2$, and $v \in M_1$ and $w \in M_2$.

If $\mathcal{X}^\alpha$ and $\mathcal{X}^\beta$ are compatible, then let $Q \subseteq [p]$ be the set of indices such that for all $q \in Q$ (ii) holds and for all $q \in [p] \backslash Q$ (i) holds. We say that $\mathcal{X}^\alpha$ and $\mathcal{X}^\beta$ are *share-compatible* if $|Q| \geq 2$. We claim that

$$T[\alpha, \mathcal{X}^\alpha] = \min_{\mathcal{X}^\beta \text{ compatible with } \mathcal{X}^\alpha} \left(T[\beta, \mathcal{X}^\beta] + \begin{cases} 1, & \text{if } \mathcal{X}^\beta \text{ and } \mathcal{X}^\alpha \text{ are share-compatible,} \\ 0, & \text{otherwise.} \end{cases} \right)$$

In other words, two signatures $\mathcal{X}^\alpha$ for node α and $\mathcal{X}^\beta$ for node β are compatible if and only if for all $q \in [p]$, either by (i) it holds that the segmentation q in $\mathcal{X}^\alpha$ is equal to the segmentation q of $\mathcal{X}^\beta$, or by (ii) it holds that the segmentation q of $\mathcal{X}^\alpha$ is the result of merging two segments in the segmentation q of $\mathcal{X}^\beta$, where none of the two segments is the zero-segment, and vertex v is in the one segment, and vertex w is in the other segment. This corresponds to connecting two trees by edge e in the forest with index q, where v is in the one tree and w in the other tree. Note that connecting two vertex-disjoint trees by exactly one edge yields a tree. The deletion of edge e in every forest of a partial solution for G_α that includes the edge e yields a partial solution for G_β. We remark that $G_\alpha = G_\beta + \{e\}$, that is, G_α differs from G_β only by the additional edge e.

Running time. For each signature $\mathcal{X}^\alpha$, we check all signatures $\mathcal{X}^\beta$ for node β for compatibility, that means, we need to check for each $q \in [p]$ whether the segmentations are equal (i) or whether the segmentation q of $\mathcal{X}^\alpha$ is derived by merging two segments in the segmentation q of $\mathcal{X}^\beta$ (ii). To check condition (i) as well as to check condition (ii) can be done in $O(p \cdot |B_\alpha|^2)$ time. Therefore, the overall running time for filling all entries in T for an introduce edge node is in $O(p \cdot (\omega + 2)^{2 \cdot p \cdot (\omega+1)+2})$.

The bottleneck in computing the tables is in the join nodes; they induce a running time portion of $O(p \cdot (\omega + 2)^{3 \cdot p \cdot (\omega+1)+3})$. Hence, filling the tables for each node in the tree decomposition can be done in the running time claimed by Theorem 1. By the above arguments about partial solutions, the minimum number of shared edges in a (p, s, t)-routing can then be read off from the table in the root node of the tree decomposition, where we take the minimum value over all signatures where for each of the p segmentations there exists a segment that contains both vertices s and t. Hence, Theorem 1 follows.

We remark that we can modify the dynamic program in such a way that we can solve the weighted variant of Minimum Shared Edges, that is, with weights $w : E(G) \to \mathbb{N}$ on the edge set of the input graph.

4 Fixed-Parameter Tractability with Respect to the Number of Paths

In this section we outline a proof for the following classification result.

▶ **Theorem 2.** Minimum Shared Edges *is fixed-parameter tractable with respect to the number p of routes.*

The basic idea for the proof is to use treewidth reduction [17], a way to process a graph G containing terminals s, t in such a way that each minimal s-t separator of size at most $p-1$ is preserved and the treewidth of the resulting graph is bounded by a function of p. The reason that this approach works is that each (p, s, t)-routing is characterized by its shared edges, and these are contained in minimal cuts of size at most $p-1$. However, treewidth reduction preserves only minimal separators, that is, vertex sets, and not necessarily minimal cuts,

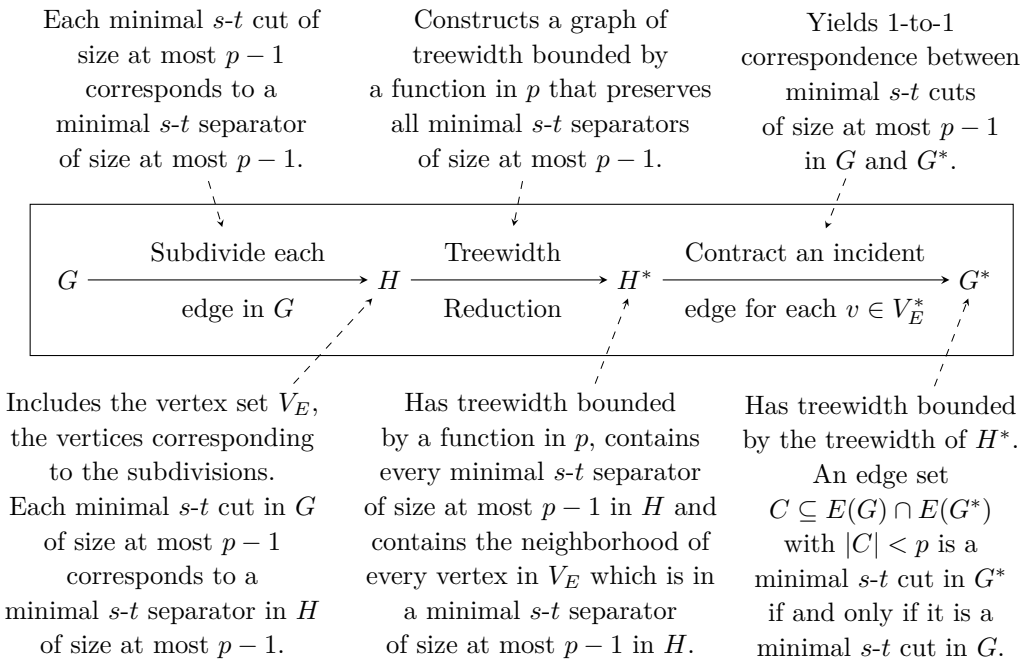

Figure 1 Overview of the strategy behind the proof of Theorem 2.

that is, edge sets. Hence, we need to further process the input graph and the graph coming out of the treewidth reduction process.

We now describe the approach in more detail; refer to Figure 1 for an overview of the following modifications and the graphs obtained in each step. In the following, we modify step by step graph G to graph G^*. We start with the following lemma which states that if our instance is a yes-instance, then we can find a solution where each of the shared edges is part of a minimal s-t cut of size smaller than the number p of routes.

▶ **Lemma 3.** *If (G, s, t, p, k) is a yes-instance of* MSE *and G has a minimal s-t cut of size smaller than p, then there exists a solution $F \subseteq E$ such that each $e \in F$ is in a minimal s-t cut of size smaller than p in G.*

Recall that if G does not have a minimal s-t cut of size smaller than p, then we can find p s-t routes without sharing an edge.

As mentioned before, as part of our approach we use the treewidth reduction technique [17]. Given a graph $G = (V, E)$ with $T = \{s, t\} \subseteq V(G)$ and an integer $\ell \in \mathbb{N}$, first the treewidth reduction technique computes the set C of vertices containing all vertices in G which are part of a minimal s-t separator of size at most ℓ in G. Then, it constructs the so-called *torso* of graph G given C and T, that is, the induced subgraph $G[C \cup T]$ with additional edges between each pair of vertices $v, w \in C \cup T$ with $\{v, w\} \notin E(G)$ if there is a v-w path in G whose internal vertices are not contained in $C \cup T$. Finally, each of these additional edges is subdivided and ℓ additional copies of each of that subdivisions are introduced, that is, if $\{v, w\}$ is one of these additional edges, then the vertices $x_1^{vw}, \ldots, x_{\ell+1}^{vw}$ are added and edge $\{v, w\}$ is replaced by the edges $\{\{v, x_1^{vw}\}, \ldots, \{v, x_{\ell+1}^{vw}\}, \{x_1^{vw}, w\}, \ldots, \{x_{\ell+1}^{vw}, w\}\}$. In the following, we denote these paths by *copy paths*. The resulting graph contains all minimal s-t separators of size at most ℓ in G and has treewidth upper-bounded by $h(\ell)$ for some function h only depending on ℓ.

▶ **Theorem 4** (Treewidth reduction [17, Theorem 2.15]). *Let G be a graph, $T \subseteq V(G)$, and let ℓ be an integer. Let C be the set of all vertices of G participating in a minimal s-t separator of size at most ℓ for some $s, t \in T$. For every fixed ℓ and $|T|$, there is a linear-time algorithm that computes a graph G^* having the following properties:*

1. *$C \cup T \subseteq V(G^*)$.*
2. *For every $s, t \in T$, a set $L \subseteq V(G^*)$ with $|L| \leq \ell$ is a minimal s-t separator of G^* if and only if $L \subseteq C \cup T$ and L is a minimal s-t separator of G.*
3. *The treewidth of G^* is at most $h(\ell, |T|)$ for some function h.*
4. *$G^*[C \cup T]$ is isomorphic to $G[C \cup T]$.*

For finding a p-routing we are interested in minimal s-t cuts of size smaller than p in G. The treewidth reduction technique guarantees to preserve minimal s-t separators of a specific size, but does not guarantee to preserve minimal s-t cuts of a specific size. Thus, we need to modify our graph G in such a way that each minimal s-t cut in G corresponds to a minimal s-t separator in the modified graph. We modify graph G in the following way.

▶ **Step 1.** *Subdivide each edge in $E(G)$, that is, for each edge $e = \{v, w\}$ in $E(G)$ add a new vertex x_e and replace edge e by edge $\{v, x_e\}$ and edge $\{x_e, w\}$. We say that vertex x_e as well as edges $\{v, x_e\}$ and $\{x_e, w\}$* correspond *to edge e. Let $V_E := \{x_e \mid e \in E\}$ and E' be the edge set replacing the edges in E. Then $H := (V \cup V_E, E')$ is the resulting graph.*

Note that each edge in H is incident with exactly one vertex in V_E and one vertex in V. Thus, no two vertices in V_E and no two vertices in V are neighbors. Moreover, note that each vertex in V_E has degree exactly two. It holds that $|V \cup V_E| = |V| + |E|$ and $|E'| = 2 \cdot |E|$.

Recall that we are interested in s-t cuts in G. By our modification from Step 1 of G to H, for each edge in G there is a corresponding vertex in V_E in H. One can show that there is a one-to-one correspondence between s-t cuts in G and those s-t separators in H that contain only vertices in V_E. Moreover, the following lemma holds.

▶ **Lemma 5.** *Every minimal s-t cut in G corresponds to a minimal s-t separator in H.*

Next, we show that every vertex in the neighborhood of each minimal s-t separator containing only vertices in V_E belongs to a minimal s-t separator.

▶ **Lemma 6.** *Let $W \subseteq V_E \subseteq V(H)$ be the set of vertices corresponding to a minimal s-t cut of size at most $\ell \in \mathbb{N}$ in G. Then, each vertex in $N_H[W]$ is part of a minimal s-t separator of size at most ℓ in H.*

We obtained graph H from graph G by applying Step 1. By Theorem 5, we know that each minimal s-t cut in G corresponds to a minimal s-t separator in H. Moreover, by Theorem 6, if we consider a minimal s-t cut of size smaller than p in G, then, for each neighbor of the vertex set in H corresponding to the minimal s-t cut in G, there exists a minimal s-t separator of size smaller than p in H that contains that neighbor. As the next step (cf. Figure 1) we apply the treewidth reduction technique [17] to graph H.

▶ **Step 2.** *Apply the treewidth reduction (Theorem 4) to graph H with $T = \{s, t\}$ and $p - 1$ as upper bound for the size of the minimal s-t separators. Denote the resulting graph by H^*.*

Let $V_E^* := \{v \in V(H^*) \mid v \in V_E\}$. Graph H^* contains all minimal s-t separators of size at most $p - 1$ in H. By Theorem 5, every minimal s-t cut of size at most $p - 1$ in G corresponds to a minimal s-t separator of size at most $p - 1$ in H and thus, by Theorem 4, to a minimal s-t separator of size at most $p - 1$ in H^*. By Theorem 6, the neighborhood of

each vertex in H corresponding to a vertex in V_E^* is contained in the vertex set $V(H^*)$. As a consequence, we can reconstruct each edge in graph G that appears in a minimal s-t cut of size at most $p-1$ in G as an edge in the graph H^*. As our next step (cf. Figure 1), we contract for each vertex in V_E^* an incident edge in graph H^*. We remark that if x^{vw} is a vertex in V_E^*, then the only edges incident with vertex x^{vw} are $\{v, x^{vw}\}$ and $\{x^{vw}, w\}$. In addition, the vertices v and w are the only neighbors of x^{vw} in graph H and in graph H^*.

▶ **Step 3.** *Contract for each vertex in V_E^* exactly one incident edge in H^* to obtain the graph G^*. In other words, undo the subdivision applied on G to obtain H.*

We remark that $\mathsf{tw}(G^*) \leq \mathsf{tw}(H^*)$, since edge contraction does not increase the treewidth of a graph [22].

Let $e = \{v, w\} \in E(G)$ be an edge in G and $x_e \in V_E \subseteq V(H)$ be the corresponding vertex in H. Then $\{v, x_e\}$ and $\{x_e, w\}$ are the incident edges of x_e in H. If $x_e \in V(H^*)$, then one of the incident edges $\{v, x_e\}$ and $\{x_e, w\}$ with vertex x_e is contracted and yields edge $\{v, w\} \in E(G^*)$. We say that the edges $\{v, w\} \in E(G)$ and $\{v, w\} \in E(G^*)$ correspond one-to-one, and, for example, we write $\{v, w\} \in E(G) \cap E(G^*)$.

Considering the graphs G and G^*, we remark that one can show that, given an s-t path in the one graph, one can find an s-t path in the other graph using a common set of edges in $E(G) \cap E(G^*)$. The next lemma states that each minimal s-t cut of size smaller than p in one of the graphs G and G^* is also a minimal s-t cut of size smaller than p in the other graph.

▶ **Lemma 7.** *Let $C \subseteq E(G) \cap E(G^*)$. Edge set C is a minimal s-t cut in G of size smaller than p if and only if C is a minimal s-t cut in G^* of size smaller than p.*

Recalling Theorem 3, we know that if an instance of MSE is a yes-instance, then we can find k edges such that the k edges form a solution for the instance and each of the k edges is part of a minimal s-t cut of size smaller than p in G. By Theorem 7, the graphs G and G^* have the same set of minimal s-t cuts of size smaller than p. Combining Theorem 3 and Theorem 7 leads to the following lemma.

▶ **Lemma 8.** *(G^*, s, t, p, k) is a yes-instance of* MSE *if and only if (G, s, t, p, k) is a yes-instance of* MSE*.*

By Theorem 8, we know that the instances (G^*, s, t, p, k) and (G, s, t, p, k) are equivalent for MSE. By our construction, we know that the treewidth of G^* is upper-bounded by a function only depending on the number p of routes. In addition, we know that Minimum Shared Edges is fixed-parameter tractable with respect to the number p of routes and an upper bound on the treewidth of the input graph. Thus, we are ready to prove Theorem 2.

Proof of Theorem 2. First we modify our graph $G = (V, E)$ by applying Steps 1 to 3. Let H, H^*, and G^* be the according graphs. By Theorem 4, the treewidth of H^* is upper-bounded by $h(p)$ for some function h. Since edge contractions do not increase the treewidth of a graph [22], it follows that $\mathsf{tw}(G^*) \leq \mathsf{tw}(H^*)$. By Theorem 8, the instances (G^*, s, t, p, k) and (G, s, t, p, k) are equivalent for MSE.

We know from Theorem 1 that MSE(p, ω) is fixed-parameter tractable when parameterized by the number p of routes and by an upper bound ω on the treewidth of the input graph. Since function h only depends on p and $h(p)$ is upper-bounding the treewidth of graph G^*, we can solve instance (G^*, s, t, p, k) in $f(p) \cdot O(|V(G^*)|)$ time, where f is a computable function only depending on parameter p. Since $|V(G^*)| \leq |V(G)| + p \cdot |E(G)| \leq p \cdot |G|$ and the instances (G^*, s, t, p, k) and (G, s, t, p, k) are equivalent for MSE, we can decide instance (G, s, t, p, k) in $f(p) \cdot p \cdot O(|G|)$ time, that is, in FPT-time. ◀

Using the dynamic program from Section 3 the running time of the above algorithm amounts to $O(p^2 \cdot (h(p)+4)^{3\cdot p\cdot(h(p)+3)+3} \cdot |G|)$. Using the bound $h(p) \leq 2^{O(p^2)}$ [17], we obtain a running time of $2^{p^3\cdot 2^{O(p^2)}} \cdot (n+m)$.

5 No Polynomial Kernel for the Parameter Number of Routes

In the previous section, we showed that Minimum Shared Edges is fixed-parameter tractable with respect to the number p of routes. It is well known that a problem is fixed-parameter tractable if and only if it admits a problem kernel. Of particular interest is the minimal possible size of a problem kernel. Accordingly, in this section we present the following lower bound.

▶ **Theorem 9.** Minimum Shared Edges *does not admit a polynomial-size problem kernel with respect to the number p of routes, unless NP $\subseteq$ coNP/poly.*

We prove Theorem 9 via an *OR-cross-composition* [4], that is, given ℓ instances of an NP-hard problem Q, all contained in one equivalence class of a polynomial-time computable relation $\mathcal{R}$ of our choosing, we compute in polynomial-time an instance (G, s, t, p, k) of MSE such that

(i) p is bounded by a polynomial function of the size of the largest input instance (*boundedness*), and

(ii) (G, s, t, p, k) is a yes-instance if and only if one of the input instances is a yes-instance (*correctness*).

If this is possible, then MSE does not admit a polynomial-size problem kernel with respect to p unless NP $\subseteq$ coNP/poly [4].

It is tempting to use MSE itself as the problem Q, to assume that each of the instances asks for the same number of routes and same number of shared edges by virtue of $\mathcal{R}$, and to OR-cross-compose by simply gluing the graphs in a chain-like fashion on sinks and sources. This fulfills the boundedness constraint, but not necessarily the correctness constraint, since the instances can "share" shared edges between them. Hence, we use the following problem as the problem Q instead.

Almost Minimum Shared Edges (AMSE)
Input: An undirected graph G, two distinct vertices $s, t \in V(G)$, and two integers $p, k \in \mathbb{N}$ such that G has a (p, s, t)-routing with at most $k+1$ shared edges.
Question: Is there a (p, s, t)-routing in G with at most k shared edges?

▶ **Proposition 10.** Almost Minimum Shared Edges *is NP-hard.*

Proposition 10 can be proven via a reduction from MSE to AMSE that introduces an additional path of length $k+1$ connecting s and t.

If we OR-cross-compose from AMSE instead, we know that if the resulting instance has a p-routing with $\ell(k+1)-1$ shared edges, then without loss of generality each of the original instances contributes at most $k+1$ shared edges. This means that at least one of the original instances is a yes-instance, giving the correctness of the OR-cross-composition.

6 W[1]-hardness with Respect to Treewidth

In this section, we present the following result.

▶ **Theorem 11.** Minimum Shared Edges *is W[1]-hard when parameterized by treewidth and the number k of shared edges combined.*

To prove Theorem 11, we give a parameterized reduction from the following problem. Herein, $\dot\cup$ denotes the disjoint union of sets.

Multicolored Clique (MCC)
Input: An undirected, k-partite graph $G = (V = V_1 \dot\cup \ldots \dot\cup V_k, E)$ with $k \in \mathbb{N}$.
Question: Is there a set $C \subseteq V$ of vertices such that $G[C]$ is a k-clique in G?

MCC is W[1]-complete when parameterized by k [11]. In the remainder of the section (G, k) is an arbitrary but fixed instance of MCC. We denote $|V_i| =: n_i$ and $V_i =: \{v_1^i, \ldots, v_{n_i}^i\}$ for all $i \in [k]$. We also say that G has the *color classes* $1, \ldots, k$, where each color class i is represented by the vertices in V_i. We write $E_{i,j} := \{\{v, w\} \in E \mid v \in V_i, w \in V_j\}$ for the edges connecting vertices in V_i and V_j, $i, j \in [k]$.

The reduction is based on the following idea. The routes we are to allocate will be split evenly into contingents of routes for each color class by a simple gadget. For each of the color classes, we introduce a selection gadget, that contains vertices (*outputs*) that correspond to the vertices in the MCC instance. Each selection gadget will route almost all the routes in its contingent to exactly one of its outputs. The outputs will then disperse $(k-1)$-times a number of routes corresponding to the ID of the vertex that this output represents. In this way, the selection gadgets represent a choice of vertices, one for each color class. In order to verify that the choice represents a clique, we introduce validation gadgets, corresponding to the pairs of color classes. They will receive the routes from the outputs of the selection gadgets, that is, the "input" of the validation gadgets is a sum of two IDs. They induce a small number of shared edges only if the vertices according to the number of routes are connected. In order to achieve this, we ensure that the sum of two IDs uniquely identifies the vertices. We achieve this by using Sidon sets.

Vertex IDs based on Sidon sets. A *Sidon set* is a set $S \subseteq \mathbb{N}$ that fulfills that for each $i, j, k, \ell \in S$ holds that if $i + k = j + \ell$ then $\{i, k\} = \{j, \ell\}$. That is, the sum of any two distinct elements in S is unique. A Sidon set S with $\max_{i \in S} i \in O(|S|^3)$ can be constructed on $O(|S|)$ time [9, page 42]. As mentioned, we use a Sidon set to distinguish numbers of routes corresponding to vertices. For this purpose, we fix a Sidon set S with $|S| = |V|$ and assign to each vertex $v \in V$ an *ID* $g(v) \in S$ where g is a bijection. For technical reasons, we need the following additional properties of g (and S):

(i) $g(v) \geq n^3$ for all $v \in V$,
(ii) $|g(v) - g(w)| \geq n^3$ for all $v, w \in V$, $v \neq w$, and
(iii) $|(g(v) + g(w)) - (g(x) + g(y))| \geq n^3$ for all $v, w, x, y \in V$, $v \neq w$, $y \notin \{v, w, x\}$.

Clearly, by adding one to each integer in the Sidon set S and then multiplying each integer by n^3 we obtain a Sidon set and a mapping g that fulfill all of the above properties simultaneously.

To enforce that only adjacent vertices are chosen in the selection gadgets, a part in a validation gadget that represents an edge must have the property that, if many routes are routed through it, then the number of routes corresponds to precisely the sum of IDs of the endpoints of the edge that is represented by this part. To do this, we have to enforce both upper and lower bounds on the sum of IDs. Upper bounds will be enforced by long parallel paths; for lower bounds, we use the notion of "complement" of an ID. For this, we define $\overline{g(v)} := M - g(v)$ for all $v \in V$, where $M := n^3 + \max_{v \in V} g(v)$. Note that $g(v) + g(w) < g(x) + g(y)$ if and only if $\overline{g(v)} + \overline{g(w)} > \overline{g(x)} + \overline{g(y)}$ for $v, w, x, y \in V$.

Construction

In the following, we describe the construction of the instance (G', s, t, p, k') of MSE, given instance (G, k) of MCC. Initially, G' consists only of the two vertices s and t, the source and the sink vertex, respectively. We describe the gadgets we use and their interconnections, which will fully describe the construction of G'. As mentioned, our gadgetry consists of two gadget types, selection gadgets on the one hand and validation gadgets on the other hand.

Before we proceed, we fix the following notation. An *m-chain* is a P_{m+1}, i.e., a path of length m. A set of ℓ m-chains with common endpoints we call an *(ℓ, m)-bundle*. A *(q, ℓ, m)-feather* is obtained by identifying one endpoint of an (ℓ, m)-bundle with one endpoint of a q-chain. In the following, by attaching a chain, bundle, or feather H to a vertex v, we mean to identify v with an endpoint of H.

We set the number of paths $p = \left(|E| - \binom{k}{2}\right) + k \cdot ((k-1) \cdot M + 1) + n$ and the number of shared edges $k' = k \cdot k^{10} + k \cdot (k + 2(k-1)) \cdot k^5 + \binom{k}{2} \cdot 3k$.

Selection gadgets. For each color class $i \in [k]$ in the instance (G, k), we construct a selection gadget $\boxed{i}$ that selects exactly one vertex of V_i as follows.

We introduce vertex c_i corresponding to color class i in (G, k). We connect s with c_i via a $((k-1) \cdot M + n_i + 1, k' + 1)$-bundle. Each of the chains in the bundle will be in exactly one route later. We introduce the vertices $x^i_1, \ldots, x^i_{n_i}$ in G', corresponding to the vertices $v^i_1, \ldots, v^i_{n_i} \in V_i$, and we connect c_i to each of them by a k^{10}-chain. These vertices serve as hubs for the routes later; only one of them will carry almost all routes in any solution, representing the choice of a vertex into the clique.

In order to relay this choice to all the validation gadgets, we do the following. First, we attach a k-chain to each vertex x^i_j, $1 \leq j \leq n_i$. Let $x^i_{j,1}, \ldots, x^i_{j,k}$ denote the vertices on the chain attached to x^i_j, indexed by the distance on the chain to vertex x^i_j; each vertex except $x^i_{j,k}$ will make its own connection to the validation gadgets. We connect each $x^i_{j,\ell}$, $\ell \in [k-1]$, with the vertex $c_i c_{\ell'}$ in the validation gadget $\boxed{i, \ell'}$ (introduced below), where $\ell' = \ell$ if $\ell < i$ and $\ell' = \ell + 1$ otherwise. The connection is made by attaching a $(k^5, g(v^i_j), k' + 1)$-feather to $x^i_{j,\ell}$ and $c_i c_{\ell'}$. Furthermore, to relay also the complement IDs, we connect each $x^i_{j,\ell}$, $\ell \in [k-1]$, with the vertex $\overline{c_i c_{\ell'}}$ by attaching a $(k^5, \overline{g(v^i_j)}, k' + 1)$-feather to them. We k^5-subdivide each edge on the k-chain we attached to x^i_j, that is, we replace each edge by a k^5-chain. We apply this to all edges on the path $x^i_1, \ldots, x^i_{n_i}$. This will ensure that in each color class, only the "ID relay vertices" $x^i_{j,\ell}$ corresponding to one ID will carry more than one route. Note that the only differences between the ID relay vertices are the second entries of the feathers, which depend on the corresponding values of the Sidon set. Finally, we connect vertex $x^i_{j,k}$ to t via a $(2, k' + 1)$-bundle; this vertex ensures that each k^5-chain between two vertices $x^i_{j,\ell}$ corresponding to the chosen ID is shared.

Validation gadgets. We need to check that the chosen vertices are adjacent using only their IDs. For this we encode the sums of IDs corresponding to two adjacent vertices into a bundle which has to be passed by the routes relayed from the selection gadgets. The budget will not allow to share any of the paths in this bundle. In this way, any sum of IDs has to be below a certain threshold. To get a lower bound, we also introduce bundles for sums of complement IDs of adjacent vertices. Finally, we ensure that an "ID" bundle and its "complement ID" bundle can be used simultaneously, only if they correspond to the same pair of vertices.

We now describe the construction of a validation gadget $\boxed{i,j}$, $i,j \in [k]$, $i < j$. We introduce exactly two vertices c_ic_j and $\overline{c_ic_j}$ (recall that these vertices already appeared in the description of the selection gadgets). We introduce a vertex for each edge between V_i and V_j, that is, if $\{v_y^i, v_z^j\} \in E_{i,j}$, then we introduce the vertex $x_y^i x_z^j$ in G'. We connect each $x_y^i x_z^j$ to c_ic_j by attaching a $(k, g(v_y^i) + g(v_z^j), k'+1)$-feather, we connect $x_y^i x_z^j$ to $\overline{c_ic_j}$ by attaching a $(k, \overline{g(v_y^i)} + \overline{g(v_z^j)}, k'+1)$-feather, and we connect $x_y^i x_z^j$ to the sink vertex t by attaching a k-chain. Only one of the connections to the sink will carry more than one route; hence, it will be possible to use only one pair of complementary bundles (corresponding to a pair of adjacent vertices).

For technical reasons, we need that each pair of bundles carries at least one route; this is achieved by also connecting s with c_ic_j via an $(|E_{i,j}| - 1, k'+1)$-bundle.

The correctness proof is deferred to the full version.

Upper-Bound on the Treewidth

To construct a tree decomposition of small width, we start out with a single bag A, where $A := \{s\} \cup \{t\} \cup \{c_i \mid i \in [k]\} \cup \{c_ic_j \mid 1 \leq i < j \leq k\} \cup \{\overline{c_ic_j} \mid 1 \leq i < j \leq k\}$. Note that $|A| = 2 + k + 2\binom{k}{2}$. Since all gadgets are interconnected via only vertices from the set A, in order to construct a tree decomposition for G', we can build a tree decomposition $\mathbb{T}'$ of each gadget separately, then add A to each of its bags, and then attach $\mathbb{T}'$ to the bag A we started with. Observe that each chain, bundle, and feather is a series-parallel graph. Since each gadget allows a tree-like structure where each edge corresponds to a series-parallel graph and each leaf is contained in A, we can find a tree decomposition of width at most 4 for each gadget. Hence, the treewidth of the graph G' as constructed above is upper-bounded by $2\binom{k}{2} + k + 2 + 4$.

7 Conclusion

Minimum Shared Edges (MSE) is a fundamental NP-hard network routing problem. We focused on exact solutions for the case of undirected, general graphs and provided several classification results concerning the parameterized complexity of MSE.

It is fair to say that our fixed-parameter tractability results (based on tree decompositions and the treewidth reduction technique [17]) are still far from practical relevance. Our studies indicate, however, that MSE is a natural candidate for performing a wider multivariate complexity analysis [12, 19] as well as studying restrictions to special graph classes. For instance, there is a simple search tree algorithm solving MSE in $O((p-1)^k \cdot (m+n)^2)$ time which might be useful in some applications [14]. Moreover, it can be shown that on unbounded undirected grids (without holes), due to combinatorial arguments, MSE can be decided in constant time after reading the input [14]. On the contrary, ongoing work indicates that MSE remains NP-hard when restricted to planar graphs (which might be of particular relevance when studying street networks). NP-hardness also prevails in case of graphs with maximum degree five [14].

In the known (pseudo) polynomial-time algorithms for graphs of bounded treewidth the exponents in the running time depend exponentially on the treewidth [1, 2]. It would be interesting to know whether a polynomial dependence is achievable.

A further line of future work is to study closely related problems and natural variants of MSE. For instance, can the positive results be transferred to the more general Minimum Vulnerability problem [2] (see the introductory section)? There are also some preliminary

investigations concerning the problem Short Minimum Shared Edges (with an additional upper bound on the maximum length of a route) [14]. Finally, it is natural to study "time-sharing" aspects for the shared edges, yielding a further natural variant of MSE.

References

1 Yusuke Aoki, Bjarni V. Halldórsson, Magnús M. Halldórsson, Takehiro Ito, Christian Konrad, and Xiao Zhou. The minimum vulnerability problem on graphs. In *Proc. 8th International Conference on Combinatorial Optimization and Applications (COCOA'14)*, volume 8881 of *LNCS*, pages 299–313. Springer, 2014.

2 Sepehr Assadi, Ehsan Emamjomeh-Zadeh, Ashkan Norouzi-Fard, Sadra Yazdanbod, and Hamid Zarrabi-Zadeh. The minimum vulnerability problem. In *Proc. 23rd International Symposium on Algorithms and Computation (ISAAC'12)*, volume 7676 of *LNCS*, pages 382–391. Springer, 2012.

3 René van Bevern, Andreas Emil Feldmann, Manuel Sorge, and Ondřej Suchý. On the parameterized complexity of computing balanced partitions in graphs. *Theory of Computing Systems*, 57(1):1–35, 2015.

4 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014.

5 Rajesh Chitnis, László Egri, and Dániel Marx. List H-coloring a graph by removing few vertices. In *Proc. 21st European Symposion on Algorithms (ESA '13)*, volume 8125 of *LNCS*, pages 313–324. Springer, 2013.

6 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

7 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *Proc. IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS'11)*, pages 150–159, 2011.

8 Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, 4th edition, 2010.

9 Apostolos Dimitromanolakis. Analysis of the Golomb ruler and the Sidon set problems, and determination of large, near-optimal Golomb rulers. Master's thesis, Department of Electronic and Computer Engineering, Technical University of Crete, June 2002. `http://www.cs.toronto.edu/~apostol/golomb/`.

10 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

11 Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009.

12 Michael R. Fellows, Bart M. P. Jansen, and Frances A. Rosamond. Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. *European Journal of Combinatorics*, 34(3):541–566, 2013.

13 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.

14 Till Fluschnik. The parameterized complexity of finding paths with shared edges. Master thesis, Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, 2015. `http://fpt.akt.tu-berlin.de/publications/theses/MA-till-fluschnik.pdf`.

15 Gregory Gutin, Mark Jones, and Bin Sheng. Parameterized complexity of the k-arc Chinese postman problem. In *Proc. 22nd European Symposium on Algorithms (ESA '14)*, volume 8737 of *LNCS*, pages 530–541. Springer, 2014.

16 Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

17 Dániel Marx, Barry O'Sullivan, and Igor Razgon. Finding small separators in linear time via treewidth reduction. *ACM Transactions on Algorithms*, 9(4):30, 2013.

18 Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2006.

19 Rolf Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In *Proc. 27th International Symposium on Theoretical Aspects of Computer Science (STACS '10)*, volume 5 of *LIPIcs*, pages 17–32. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2010.

20 Masoud T. Omran, Jörg-Rüdiger Sack, and Hamid Zarrabi-Zadeh. Finding paths with minimum shared edges. *Journal of Combinatorial Optimization*, 26(4):709–722, 2013.

21 Andrzej Pelc. Fault-tolerant broadcasting and gossiping in communication networks. *Networks*, 28(3):143–156, 1996.

22 Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

23 Douglas B. West. *Introduction to Graph Theory.* Prentice Hall, 2 edition, 2000.

24 Zhi-Qian Ye, Yi-Ming Li, Hui-Qiang Lu, and Xiao Zhou. Finding paths with minimum shared edges in graphs with bounded treewidths. In *Proc. Frontiers of Computer Science (FCS'13)*, pages 40–46, 2013.