On the Parameterized Complexity of Eulerian Strong Component Arc Deletion

Václav Blažej
University of Warwick, Coventry, UK

Satyabrata Jana
University of Warwick, Coventry, UK

M. S. Ramanujan
University of Warwick, Coventry, UK

Peter Strulo
University of Warwick, Coventry, UK

Abstract

In this paper, we study the Eulerian Strong Component Arc Deletion problem, where the input is a directed multigraph and the goal is to delete the minimum number of arcs to ensure every strongly connected component of the resulting digraph is Eulerian.

This problem is a natural extension of the Directed Feedback Arc Set problem and is also known to be motivated by certain scenarios arising in the study of housing markets. The complexity of the problem, when parameterized by solution size (i.e., size of the deletion set), has remained unresolved and has been highlighted in several papers. In this work, we answer this question by ruling out (subject to the usual complexity assumptions) a fixed-parameter tractable (FPT) algorithm for this parameter and conduct a broad analysis of the problem with respect to other natural parameterizations. We prove both positive and negative results. Among these, we demonstrate that the problem is also hard (W[1]-hard or even para-NP-hard) when parameterized by either treewidth or maximum degree alone. Complementing our lower bounds, we establish that the problem is in XP when parameterized by treewidth and FPT when parameterized either by both treewidth and maximum degree or by both treewidth and solution size. We show that these algorithms have near-optimal asymptotic dependence on the treewidth assuming the Exponential Time Hypothesis.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Parameterized complexity, Eulerian graphs, Treewidth

Digital Object Identifier 10.4230/LIPIcs.IPEC.2024.4

Related Version *Full Version*: `https://arxiv.org/abs/2408.13819` [2]

Funding Supported by the Engineering and Physical Sciences Research Council (grant number EP/V044621/1).

1 Introduction

In the Eulerian Strong Component Arc Deletion (ESCAD) problem, where the input is a directed graph (digraph)[1] and a number k and the goal is to delete at most k arcs to ensure every strongly connected component of the resulting digraph is Eulerian. This problem was

[1] In this paper, the arc set of a digraph is a multiset, i.e., we allow multiarcs. Moreover, we treat multiarcs between the same ordered pairs of vertices as distinct arcs in the input representation of all digraphs. Consequently, the number of arcs in the input is upper bounded by the length of the input. We exclude loops as they play no non-trivial role in instances of this problem.

first introduced by Cechlárová and Schlotter [3] to model problems arising in the study of housing markets and they left the existence of an FPT algorithm for ESCAD as an open question.

The ESCAD problem extends the well-studied Directed Feedback Arc Set (DFAS) problem. In DFAS, the goal is to delete the minimum number of arcs to make the digraph acyclic. The natural extension of DFAS to ESCAD introduces additional complexity as we aim not to prevent cycles, but aim to balance in-degrees and out-degrees within each strongly connected component. As a result, the balance requirement complicates the problem significantly and the ensuing algorithmic challenges have been noted in multiple papers [3, 6, 11].

Crowston et al. [4] made partial progress on the problem by showing that ESCAD is fixed-parameter tractable (FPT) on tournaments and also gave a polynomial kernelization. However, the broader question of fixed-parameter tractability of ESCAD on general digraphs has remained unresolved.

Our contributions. Our first main result rules out the existence of an FPT algorithm for ESCAD under the solution-size parameterization, subject to standard complexity-theoretic assumptions.

▶ **Theorem 1.** *ESCAD is* W[1]*-hard parameterized by the solution size.*

The above negative result explains, in some sense, the algorithmic challenges encountered in previous attempts at showing tractability and shifts the focus toward alternative parameterizations. However, even here, we show that a strong parameterization such as the vertex cover number is unlikely to lead to a tractable outcome.

▶ **Theorem 2.** *ESCAD is* W[1]*-hard parameterized by the vertex cover number of the graph.*

In fact, assuming the Exponential Time Hypothesis (ETH), we are able to obtain a stronger lower bound.

▶ **Theorem 3.** *There is no algorithm solving ESCAD in $f(k) \cdot n^{o(k/\log k)}$ time for some function f, where k is the vertex cover number of the graph and n is the input length, unless the Exponential Time Hypothesis fails.*

To add to the hardness results above, we also analyze the parameterized complexity of the problem parameterized by the maximum degree of the input digraph and show that even for constant values of the parameter, the problem remains NP-hard.

▶ **Theorem 4.** *ESCAD is* NP*-hard in digraphs where each vertex has* $(\mathrm{in}, \mathrm{out})$ *degrees in* $\{(1, 6), (6, 1)\}$.

We complement these negative results by showing that ESCAD is FPT parameterized by the treewidth of the deoriented digraph (i.e., the underlying undirected multigraph) and solution size as well as by the treewidth and maximum degree of the input digraph. Furthermore, we give an XP algorithm parameterized by treewidth alone. All three results are obtained by a careful analysis of the same algorithm stated below.

▶ **Theorem 5.** *An ESCAD instance $\mathcal{I} = (G, k)$ can be solved in time $2^{\mathcal{O}(\mathtt{tw}^2)} \cdot (2\alpha+1)^{2\mathtt{tw}} \cdot n^{\mathcal{O}(1)}$ where $\mathtt{tw}$ is the treewidth of deoriented G, Δ is the maximum degree of G, and $\alpha = \min(k, \Delta)$.*

In the above statement, notice that α is upper bounded by the number of edges in the digraph and so, implies an XP algorithm parameterized by the treewidth with running time $2^{\mathcal{O}(\mathtt{tw}^2)} \cdot n^{\mathcal{O}(\mathtt{tw})}$. Notice the running time of our algorithm asymptotically almost matches our ETH based lower bound (recall that the vertex cover number of a graph is at least the treewidth) in Theorem 3.

Recall that in general, multiarcs are permitted in an instance of ESCAD. This fact is crucially used in the proof of Theorem 2 and raises the question of adapting this reduction to *simple digraphs* (digraphs without multiarcs or loops) in order to obtain a similar hardness result parameterized by vertex cover number. However, we show that this is not possible by giving an FPT algorithm for the problem on simple digraphs parameterized by the vertex integrity of the input graph. Recall that a digraph has *vertex integrity* k if there exists a set of vertices of size $q \leq k$ which when removed, results in a digraph where each weakly connected component has size at most $k - q$. Vertex integrity is a parameter lower bounding vertex cover number and has gained popularity in recent years as a way to obtain FPT algorithms for problems that are known to be W[1]-hard parameterized by treedepth – one example being ESCAD on simple graphs as we show in this paper (see Theorem 7 below).

▶ **Theorem 6.** *ESCAD on simple digraphs is* FPT *parameterized by the vertex integrity of the graph.*

As a consequence of this result, we infer an FPT algorithm for ESCAD on simple digraphs parameterized by the vertex cover number, highlighting the difference in the behaviour of the ESCAD problem on directed graphs that permit multiarcs versus simple digraphs. On the other hand, we show that even on simple digraphs this positive result does not extend much further to well-studied width measures such as treewidth (or even the larger parameter treedepth), by obtaining the following consequence of Theorems 2 and 3.

▶ **Theorem 7.** *ESCAD even on simple digraphs is* W[1]*-hard parameterized by k and assuming* ETH*, there is no algorithm solving it in $f(k)n^{(k/\log k)}$ time for some function f, where k is the size of the smallest vertex set that must be deleted from the input digraph to obtain a disjoint union of directed stars and n is the input length.*

Related Work. The vertex-deletion variant of ESCAD is known to be W[1]-hard, as shown by Göke et al. [11], who identify ESCAD as an open problem and note that gaining more insights into its complexity was a key motivation for their study. Cygan et al. [6] gave the first FPT algorithm for edge (arc) deletion to Eulerian graphs (respectively, digraphs). Here, the aim is to make the whole graph Eulerian whereas the focus in ESCAD is on each strongly connected component. Cygan et al. also explicitly highlight ESCAD as an open problem and a motivation for their work. Goyal et al. [12] later improved the algorithm of Cygan et al. by giving algorithms achieving a single-exponential dependence on k.

2 Preliminaries

For a digraph G, we denote its vertices by $V(G)$, arcs by $E(G)$, the subgraph induced by $S \subseteq V(G)$ as $G[S]$, a subgraph with subset of vertices removed as $G - S = G[V(G) \setminus S]$, and a subgraph with subset of edges $F \subseteq E(G)$ removed as $G - F = (V(G), E(G) \setminus F)$. For a vertex v and digraph G, let $\deg^-_G(v)$ denote its in-degree, $\deg^+_G(v)$ be its out-degree, and $\deg^+_G(v) - \deg^-_G(v)$ is called its *imbalance*. If the imbalance of v is 0 then v is said to be *balanced* (in G). A digraph is called *balanced* if all its vertices are balanced. The maximum degree of a digraph G is the maximum value of $\deg^+_G(v) + \deg^-_G(v)$ taken over every vertex v in the graph.

A vertex v is *reachable* from u if there exists a directed path from u to v in G. A *strongly connected component* of G is a maximal set of vertices where all vertices are mutually reachable. Let *strong subgraph* denoted $\text{strong}(G)$ be the subgraph of G obtained by removing all arcs that have endpoints in different strongly connected components. The ESCAD problem can now be formulated as "Is there a set $S \subseteq V(G)$ of size $|S| \leq k$ such that $\text{strong}(G - S)$ is balanced?" We call an arc $e \in E(G)$ *active* in G if $e \in E(\text{strong}(G))$ and *inactive* in G otherwise.

A graph G has vertex cover k if there exists a set of vertices $S \subseteq V(G)$ with bounded size $|S| \leq k$ such that $G - S$ is an independent set. A star is an undirected graph isomorphic to K_1 or $K_{1,t}$ for some $t \geq 0$ and a *directed star* is just a digraph whose underlying undirected graph is a star.

A *tree decomposition* of an undirected graph G is a pair $(T, \{X_t\}_{t\in V(T)})$ where T is a tree and $X_t \subseteq V(G)$ such that (i) for all edges $uv \in E(G)$ there exists a node $t \in V(T)$ such that $\{u, v\} \subseteq X_t$ and (ii) for all $v \in V(G)$ the subgraph induced by $\{t \in V(T) : v \in X_t\}$ is a non-empty tree. The *width* of a tree decomposition is $\max_{t\in V(T)} |X_t| - 1$. The *treewidth* of G is the minimum width of a tree decomposition of G.

Let $(T, \{X_t\}_{t\in V(T)})$ be a tree decomposition of G. We refer to every node of T with degree one as a *leaf node* except one which is chosen as the root, r. A tree decomposition $(T, \{X_t\}_{t\in V(T)})$ is a *nice tree decomposition with introduce edge nodes* if all of the following conditions are satisfied:

1. $X_r = \emptyset$ and $X_\ell = \emptyset$ for all leaf nodes ℓ.
2. Every non-leaf node of T is one of the following types:
   - **Introduce vertex node:** a node t with exactly one child t' such that $X_t = X_{t'} \cup v$ for some vertex $v \notin X_{t'}$.
   - **Introduce edge node:** a node t, labeled with an edge uv where $u, v \in X_t$ and with exactly one child t' such that $X_t = X_{t'}$.
   - **Forget node:** a node t with exactly one child t' such that $X_t = X_{t'} \setminus \{v\}$ for some vertex $v \in X_{t'}$.
   - **Join node:** a node t with exactly two children t_1, t_2 such that $X_t = X_{t_1} = X_{t_2}$.
3. Every edge appears on exactly one introduce edge node.

Proofs for results marked $\star$ can be found in the full version [2].

3 Our Results for ESCAD

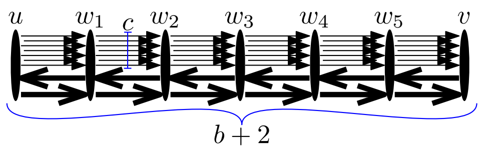
In the following four subsections we describe three hardness results and tractability results on bounded treewidth graphs for ESCAD. In Section 3.4 we show that the problem is XP by treewidth and FPT in two cases – when parameterized by the combined parameter treewidth plus maximum degree, and when parameterized by treewidth plus solution size. The hardness results show that dropping any of these parameters leads to a case that is unlikely to be FPT. More precisely, we show that parameterized by solution size it is W[1]-hard (in Section 3.1) as is the case when parameterized by vertex cover number (Section 3.2), and it is para-NP-hard when when parameterized by the maximum degree (Section 3.3).

3.1 W[1]-hardness of ESCAD Parameterized by Solution Size

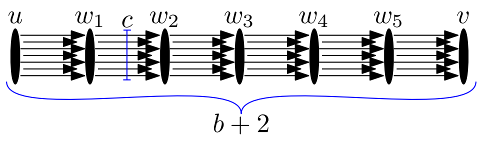
In this section, we show that ESCAD is W[1]-hard when parameterized by solution size. Our reduction is from Multicolored Clique. The input to Multicolored Clique consists of a simple undirected graph G, an integer ℓ, an containing exactly one vertex from each set $V_i, i \in [\ell]$. Multicolored Clique is known to be W[1]-hard when parameterized by the size of the solution ℓ [5]. Each set V_i for $i \in [\ell]$ is called a color class and for a vertex v in G, we say v has color i if $v \in V_i$. We assume without loss of generality that in the Multicolored Clique instance we reduce from, each color class V_i forms an independent set (edges in the same color class can be removed) and moreover, for each vertex $v \in V_i$ and each $j \in [\ell] \setminus \{i\}$ there exists a $w \in V_j$ that is adjacent to v (any vertex that cannot participate in a multicolored clique can be removed).

We start with descriptions of two auxiliary gadgets: the *imbalance gadget* and the *path gadget.*

Imbalance Gadget. Let u, v be a pair of vertices, and b, c be two positive integers. We construct a gadget $I_{u,v}$ connecting the vertex u to v by a path with vertices $u, w_1, \ldots, w_b, v$ where w_i's are b new vertices (we call them intermediate vertices in this gadget); let $u = w_0$ and $v = w_{b+1}$. For every $i \in \{0, \ldots, b\}$ the path contains $b+1+c$ forward arcs (w_i, w_{i+1}) and $b+1$ backward arcs (w_{i+1}, w_i), see Figure 1a for an illustration. Observe that with respect to the gadget $I_{u,v}$, the vertices u and v have imbalances c and $-c$, respectively, whereas other vertices in the gadget have imbalance zero. We refer to this gadget $I_{u,v}$ as a (b, c)-*imbalance gadget.*



(a) c-imbalance gadget.



(b) c-path gadget.

Figure 1 Black ellipses are vertices, thick edges represent $(b+1)$ copies of the edges; $(b+2)$ is the number of vertices in the gadgets.

Path Gadget. Let u, v be a pair of vertices, and b, c be two positive integers. We construct a gadget $P_{u,v}$ connecting the vertex u to v by a path with vertices $u, w_1, \ldots, w_b, v$ where w_i's are b new intermediate vertices; let $u = w_0$ and $v = w_{b+1}$. For every $i \in \{0, \ldots, b\}$ the path contains c forward arcs (w_i, w_{i+1}). See Figure 1b for an illustration. Notice that, unlike the imbalance gadget, we do not add backward arcs. Observe that with respect to the gadget $P_{u,v}$, the vertices u and v has imbalances c and $-c$, respectively, whereas the other vertices in the gadget have imbalance zero. We refer to this gadget $P_{u,v}$ as a (b, c)-*path gadget.*

We use the following properties of the gadgets I_{uv} and P_{uv} to reason about the correctness of our construction.

▶ **Lemma 8** ($\star$). *Let (G, b) be a yes-instance of ESCAD and S be a solution. Assume that for a pair of vertices u, v in G, there is a (b, c)-imbalance gadget I_{uv} present in G (i.e., I_{uv} is an induced subgraph of G). If S is an inclusionwise minimal solution then S contains no arc of I_{uv}.*

▶ **Lemma 9** ($\star$). *Let (G, b) be a yes-instance of ESCAD and S be an inclusionwise minimal solution for this instance. Assume that for a pair of vertices u, v in G, there is a (b, c)-path gadget P_{uv} present in G (i.e., P_{uv} is an induced subgraph of G) and there are more than b arc-disjoint paths from v to u. If S contains an arc from P_{uv}, then there exists $i \in \{0, \ldots, b+1\}$ such that S contains every (w_i, w_{i+1}) arc in P_{uv}.*

Brief idea of the reduction. The main idea of the following reduction is to "choose" vertices and edges of the clique using cuts. First, we enforce an imbalance using (b, c)-imbalance gadgets where b is the budget and let it propagate using path gadgets in a way that chooses a vertex for each color. For each chosen vertex, the solution is then forced to select $(\ell - 1)$ out-going arcs that are incident to it. Choosing the same edge from two sides results in a specific vertex to be cut from the strongly connected component of the remaining graph, decreasing the degree by the correct amount. Our solution creates a set of $\binom{\ell}{2}$ vertices that have out-degree two – these vertices represent edges of the multicolored clique.

▶ **Theorem 1.** *ESCAD is* W[1]*-hard parameterized by the solution size.*

Proof. Consider an instance $\mathcal{I} = \big(G, \ell, (V_1, \dots, V_\ell)\big)$ of Multicolored Clique with n vertices. Recall our assumption that each color class induces an independent set, and every vertex has at least one neighbor in every color class distinct from its own. In polynomial time, we construct an ESCAD instance $\mathcal{I}' = (G', k)$ in the following way (see Figure 2 for an overview).

- We set $k = 2\ell(\ell - 1)$.
- Construction of $V(G')$ is as follows:
  1. We add a vertex s.
  2. For each color $j \in [\ell]$, we have a pair of vertices s_j and d_j.
  3. For each vertex u in $V(G)$, we have a vertex x_u.
  4. For each edge uv in $E(G)$, we have a vertex z_{uv}.
- The construction of $E(G')$ is as follows. We introduce four sets of arcs E_1, E_2, E_3, and E_4 that together comprise the set $E(G')$. For each color $j \in [\ell]$, let $r_j \coloneqq |V_j| \cdot (\ell - 1)$, $c_j \coloneqq |\{uv : uv \in E(G), u \in V_j\}| - r_j$. Notice that $c_j \geq 0$ since every vertex in G has degree at least $\ell - 1$.
  1. For each $j \in [\ell]$, we add a $(k, r_j - \ell + 1)$-imbalance gadget I_{s,d_j} and a (k, c_j)-imbalance gadget I_{s,s_j} to E_1.
  2. For each $j \in [\ell]$, for each vertex $u \in V_j$ we add a $(k, |N_G(u)| - \ell + 1)$-imbalance gadget I_{s_j,x_u} and a $(k, \ell - 1)$-path gadget P_{d_j,x_u} to E_2.
  3. For every edge $uv \in E(G)$, we add a pair of arcs (x_u, z_{uv}) and (x_v, z_{uv}) to E_3.
  4. For every edge $uv \in E(G)$, we add two copies of the arc (z_{uv}, s) to E_4.

It is easy to see that the construction can be performed in time polynomial in $|V(G)|$. Now, we prove the correctness of our reduction. First, we argue about the imbalances of vertices in G'. As each vertex of G' lies on a cycle that goes through s, it follows that G' is strongly connected.

▷ Claim 10 (⋆). The only vertices with non-zero imbalance in G' are those in the set $\{s\} \cup \{d_j : j \in [\ell]\}$. Furthermore, the imbalance of the vertex s is $-\ell(\ell-1)$ and the imbalance of d_j for each $j \in [\ell]$ is $(\ell - 1)$.

This shows that there are only $\ell+1$ vertices with non-zero imbalance in G'. The imbalance of the d_j's will make us "choose" vertices and edges that represent a clique in G as we will see later.

We now show correctness of our reduction. In the forward direction, assume that (G, ℓ) is a yes-instance and let K be a multicolored clique of size ℓ in G. Let v_j denote the vertex with color j in K. We now construct a solution S of (G', k). For each edge $v_i v_j$ we add the arcs $(x_{v_j}, z_{v_j v_i})$ and $(x_{v_i}, z_{v_j v_i})$ to S. There are $2 \cdot \binom{\ell}{2}$ many such arcs. Now for each $j \in [\ell]$ we add all the incoming arcs of x_{v_j} along the path gadget $P_{d_j x_{v_j}}$ to S. As for each $j \in [\ell]$,

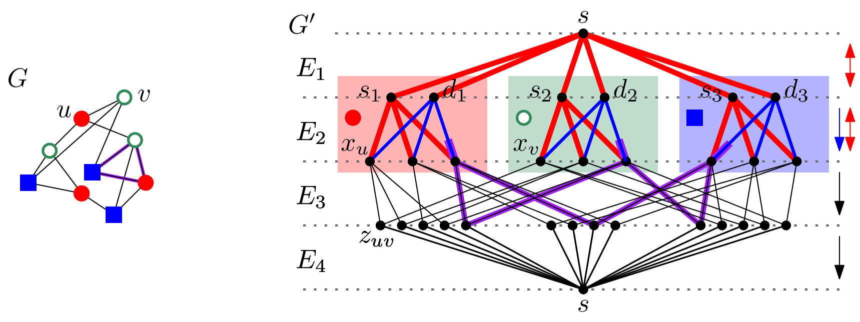


Figure 2 Overview of the reduction from G to G'; four sets of edges are depicted from top to bottom. E_1 contains imbalance gadgets, E_2 is a mixture of imbalance and path gadgets, E_3 has directed edges, and E_4 has all directed double edges to s. Marked purple edges corresponds to a solution in G and its respective solution in G'. The thee colored backgrounds in G' signify part of the construction tied to the three color classes. All edges of the picture of G' are oriented from top to bottom. The picture of G' wraps up as the vertex s drawn on the bottom is the same as the one drawn on the top.

the number of such arcs is $(\ell-1)$ we have $|S| = 2 \cdot \binom{\ell}{2} + \ell(\ell-1) = 2\ell(\ell-1) = k$. Now, we show that each strongly connected component in $G' - S$ is Eulerian. For an example of S, refer to Figure 2 (purple arcs).

We consider the strongly connected components of $G' - S$ and we will show that each of them is Eulerian. We first define:

$$Z = \{z_{uw} : u, w \in K\} \cup \Big(\bigcup_{j\in[\ell]} (V(P_{d_j,x_{v_j}}) \setminus \{d_j, x_{v_j}\}) \Big)$$

▷ Claim 11 (⋆). One strongly connected component of $G' - S$ consists of all the vertices except Z (we call it the *large* component) and all other strongly connected components of $G' - S$ are singleton – one for each vertex in Z.

Since singleton strongly connected components are always balanced, we only need to show that the large component is Eulerian i.e., it is balanced inside the strongly connected component itself.

▷ Claim 12 (⋆). The large component is Eulerian

The claim can be proved through a case analysis going through the various types of vertices that appear in this strongly connected component, i.e., the vertex s, the vertices in $\{s_j : j \in [\ell]\} \cup \{z_{uv} : u \notin K \text{ or } v \notin K\} \cup \{x_u : u \notin K\}$, the vertex d_j for $j \in [\ell]$, and x_u where $u \in K$.

This completes the argument in the forward direction.

In the converse direction, assume that (G', k) is a yes-instance and let S be a solution. Let us first establish some structure on S, from which it will be possible to recover a multicolored clique for G.

Let $\mathcal{C}$ denote the strongly connected component of $G' - S$ that contains s. Due to Lemma 8, we may assume that S does not contain any arcs of any of the imbalance gadgets. This implies that $\mathcal{C}$ contains s_j and d_j for every $j \in [\ell]$ as well as x_u for every $u \in V(G)$. Moreover, due to Lemma 9, we know that if S contains arcs of a path gadget P_{d_j,x_u}, then they form a cut in it. As all inclusion-wise minimal cuts of the path gadgets are of the same cardinality and adding any minimal cut of a path gadget to S makes all arcs of the path gadget inactive in $G' - S$, assume that if a cut of a path gadget P_{d_j,x_u} is in S, then the cut consists of the incoming-arcs of x_u in the gadget.

Recall from Claim 10, that the only imbalanced vertices in G' are $\{s\} \cup \{d_j : j \in [\ell]\}$. Let us make some observations based on the fact that these vertices are eventually balanced in strong$(G' - S)$.

For each $j \in [\ell]$, since none of the incoming arcs of d_j are in S (they lie in an imbalance gadget), in order to make d_j balanced it must be the case that S contains a cut of exactly one of the path gadgets starting at d_j, call it $P_{d_j,x_{v_j}}$. Recall that x_{v_j} was originally balanced in G'. Further, recall that we have argued that x_{v_j} is in $\mathcal{C}$ along with s_j and d_j. Since the imbalance gadget starting at s_j and ending at x_{v_j} cannot intersect S and we have deleted all of the $\ell - 1$ incoming arcs to x from the path gadget $P_{d_j,x_{v_j}}$, the imbalance of $-\ell + 1$ thus created at x_{v_j} needs to be resolved by making exactly $(\ell - 1)$ of its outgoing arcs in E_3 inactive in $G' - S$. Since we have already spent a budget of $\ell(\ell - 1)$ from the path gadgets, the budget that remains to be used for resolving these imbalances at $\{x_{v_j} : j \in [\ell]\}$ is $\ell(\ell-1)$.

On the other hand, recall that s is imbalanced in G' and to make s balanced, we need to make $\ell(\ell - 1)$ incoming arcs of s (from E_4) inactive in $G' - S$. This is because all outgoing arcs of s lie in imbalance gadgets and cannot be in S.

And finally, recall that for each $uv \in E(G)$, the vertex z_{uv} is balanced in G' (by Claim 10). Since the strongly connected component $\mathcal{C}$ in $G' - S$ contains the vertices s, x_u, x_v (i.e., all neighbors of $z_{u,v}$), for the vertex z_{uv} to remain balanced in strong$(G' - S)$, we have the following exhaustive cases regarding the arcs between $s, x_u, x_v, z_{u,v}$: (1) none of the four arcs incident to z_{uv} is in S; (2) one incoming and one outgoing arc are in S; (3) both incoming arcs or both outgoing arcs are in S. In Case (2), two arcs are added to S, which makes two arcs inactive while in Case (3) two arcs are added to S which makes four arcs inactive. As previously noted, we still need $\ell(\ell-1)$ arcs in E_3 and $\ell(\ell-1)$ arcs in E_4 to become inactive in $G' - S$. The required number of inactive arcs in $E_3 \cup E_4$ is twice the remaining budget, so for every z_{uv}, x_u, x_v, the arcs between $s, x_u, x_v, z_{u,v}$ must be in Case (1) or Case (3). Moreover, whenever Case (3) occurs, we may assume without loss of generality that the arcs in S are the two arcs (x_u, z_{uv}) and (x_v, z_{uv}). Thus, there are exactly $\binom{\ell}{2}$ vertices z_{uv} such that the arcs between $s, x_u, x_v, z_{u,v}$ are in Case (3).

We now extract the solution clique K for (G, ℓ) by taking, for each $j \in [\ell]$, the vertex $v_j \in V(G)$ such that a cut of $P_{d_j,x_{v_j}}$ is contained in S. We have shown that there are exactly $\binom{\ell}{2}$ vertices z_{uv} such that the arcs between $s, x_u, x_v, z_{u,v}$ are in Case (3) and for each $j \in [\ell]$ and the vertex x_{v_j}, exactly $\ell - 1$ of its outgoing arcs are made inactive by S. This can only happen if for every $j, j' \in [\ell]$, there is a vertex $z_{v_j v_{j'}}$, implying that $v_j v_{j'}$ is an edge in G. ◄

3.2 W[1]-hardness of ESCAD Parameterized by Vertex Cover Number

In this section, we show that ESCAD is W[1]-hard when parameterized by the vertex cover number. Jansen, Kratsch, Marx, and Schlotter [13] showed that Unary Bin Packing is W[1]-hard when parameterized by the number of bins h.

Unary Bin Packing
Input: A set of positive integer item sizes $x_1, \ldots, x_n$ encoded in unary, a pair of integers h and b.
Question: Is there a partition of $[n]$ into h sets $J_1, \ldots, J_h$ such that $\sum_{\ell \in J_j} x_\ell \leq b$ for every $j \in [h]$?

In order to carefully handle vertex balances in our reduction, it is helpful to work with a variant of the above problem, called Exact Unary Bin Packing, where the inequality $\sum_{\ell \in J_j} x_\ell \leq b$ is replaced with the equality $\sum_{\ell \in J_j} x_\ell = b$. That is, in this variant, all bins get filled up to their capacity.

▶ **Theorem 2.** *ESCAD is* W[1]*-hard parameterized by the vertex cover number of the graph.*

Proof. Let $\mathcal{I}' = \big((x_1, \dots, x_m), h, b\big)$ be an instance of Unary Bin Packing. If $b \geq \sum_{i=1}^{m} x_i$, then $\mathcal{I}'$ is trivially a yes-instance and we can return a trivial yes-instance of ESCAD with vertex cover number at most h. In the same way, if $b \cdot h < \sum_{i \in [m]} x_i$, then $\mathcal{I}'$ is trivially a no-instance and we return a trivial no-instance of ESCAD with vertex cover number at most h. Now, suppose neither of the above cases occur.

Note that the length of the unary encoding of b is upper bounded by the total length of the unary encoding of all items $x_1, \dots, x_m$. Similarly, if $h \geq m$ then the instance boils down to checking whether $x_i \leq b$ for every $i \in [m]$ (and producing a trivial ESCAD instance accordingly) so we can assume that $h < m$, hence, the length of the unary encoding of h is upper bounded by the total length of the unary encoding of all items. We now construct an instance $\mathcal{I}$ of Exact Unary Bin Packing from $\mathcal{I}'$ by adding $h \cdot b - \sum_{i \in [m]} x_i$ one-sized items (this is non-negative because of the preprocessing steps). If $\mathcal{I}'$ is a yes-instance, then one can fill-in the remaining capacity in every bin with the unit-size items, to get a solution for $\mathcal{I}$. Conversely, if $\mathcal{I}$ is a yes-instance, then removing the newly added unit-size items yields a solution for $\mathcal{I}'$. Let n denote the number of items in $\mathcal{I}$. Note that since $\sum_{i \in [n]} x_i = b \cdot h$, this implies that $|\mathcal{I}| = \mathcal{O}(|\mathcal{I}'|^2)$, the instance of Exact Unary Bin Packing remains polynomially bounded.

We next reduce the Exact Unary Bin Packing instance $\mathcal{I}$ to an instance $\mathcal{I}^* = (G, k)$ of ESCAD in polynomial time. Let us fix the budget $k = b \cdot h(h-1)$. We now build a graph G that models the bins by k copies of interconnected gadgets (that form the vertex cover) and models each item as a vertex of the independent set. In our reduction, we use the following terms. For a pair of vertices p, q, a c-arc (p, q) denotes c parallel copies of the arc (p, q) and a thick arc (p, q) denotes a $3k$-arc (p, q). The construction of G is as follows.

- The vertex set of G is the set $\{u_j : j \in [h]\} \cup \{v_j : j \in [h]\} \cup \{w_i : i \in [n]\}$.
- For each $j \in [h]$, we add a b-arc (u_j, v_j), a thick arc (u_j, v_j) and a thick arc (v_j, u_j). We call the subgraph induced by u_j, v_j and these arcs, the *b-imbalance gadget B_j*.
- Next, we add thick arcs $(u_j, u_{j'})$ for every $j < j'$ where $j, j' \in [h]$.
- Finally, for each $i \in [n]$ and $j \in [h]$, we add x_i-arcs (w_i, u_j) and (v_j, w_i).

This concludes the construction, see Figure 3. Before we argue the correctness, let us make some observations.

Note that the vertices participating in the imbalance gadgets form a vertex cover of the resulting graph and their number is upper bounded by $2h$. Hence, if we prove the correctness of the reduction, we have the required parameterized reduction from Unary Bin Packing parameterized by the number of bins to ESCAD parameterized by the vertex cover number of the graph.

We say that a set of arcs in G *cuts* a (p, q) arc if it contains all parallel copies of (p, q). Note that no set of at most k arcs cuts a thick (p, q) arc. In particular, no solution to the ESCAD instance (G, k) cuts any thick arc (p, q) that appears in the graph.

Exact Unary Bin Packing is a yes-instance $\Rightarrow$ ESCAD is a yes-instance. Assume that we have a partition $J_1, \dots, J_h$ that is a solution to $\mathcal{I}$. We now define a solution S for $\mathcal{I}^*$. For every $x_i \in J_j$ we cut (i.e., add to S) all parallel copies of the arc $(w_i, u_{j'})$ for every $j' < j$ and

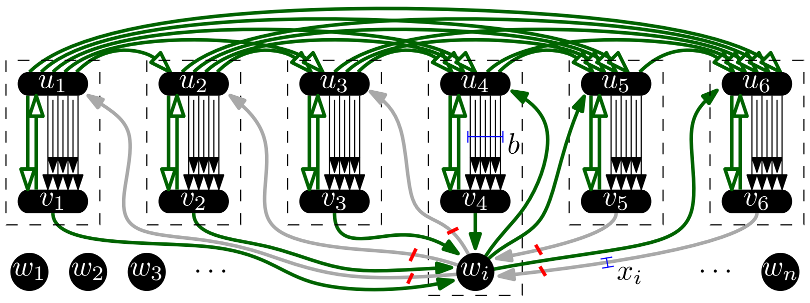


Figure 3 A part of the resulting ESCAD instance after reduction from EXACT UNARY BIN PACKING with six bins; connections between the independent vertices and imbalance gadgets are shown only for one vertex w_i. Thick arcs are shown with empty arrowhead, bold arcs incident to w_i are x_i-arcs. Crossed off arcs are in a solution and dashed boxes show strongly connected components of the solution. This example represents $x_i \in J_4$.

we cut all parallel copies of the arc $(v_{j''}, w_i)$ for every $j'' > j$. This results in cutting a total of $x_i \cdot (h-1)$ arcs incident to each w_i and as $\sum_{i=1}^{n} x_i = b \cdot h$ we cut exactly $b \cdot h(h-1) = k$ arcs in total.

▷ Claim 13 (⋆). strong$(G-S)$ is balanced.

ESCAD is a yes-instance ⇒ Exact Unary Bin Packing is a yes-instance. We aim to show that in any solution for the ESCAD instance, the arcs that are cut incident to w_i for any $i \in [n]$ have the same structure as described in the other direction, i.e., for all w_i there exists j such that the solution cuts $(w_i, u_{j'})$ for all $j' < j$ and it cuts $(v_{j''}, w_i)$ for all $j'' > j$. This is equivalently phrased in the following claim.

▷ Claim 14 (⋆). There are no two indices $a, b \in [h]$ with $a < b$ such that both (w_i, u_a) and (v_b, w_i) are uncut.

We next argue that if S is a solution, then for all w_i, there exists j such that the solution is disjoint from any (w_i, u_j) arc and any (v_j, w_i) arc. Since the budget is $k = b \cdot h(h-1)$ we have that: If we cut more than $x_i(h-1)$ arcs incident to w_i for some $i \in [n]$, then there exists $i' \in [n] \setminus \{i\}$ such that we cut fewer than $x_{i'}(h-1)$ arcs incident to $w_{i'}$. But this would violate Claim 14. Hence, for any solution S, we can retrieve the assignment of items to bins in the EXACT UNARY BIN PACKING instance $\mathcal{I}$, by identifying for every $i \in [n]$, the unique value of $j \in [h]$ such that S is disjoint from any (w_i, u_j) arc and any (v_j, w_i) arc and then assigning item x_i to bin J_j. ◀

Besides establishing that UNARY BIN PACKING does not have an FPT algorithm unless W[1] = FPT, Jansen et al. [13] showed that under the stronger assumption[2] of the Exponential Time Hypothesis (ETH) the well-known $n^{\mathcal{O}(h)}$-time algorithm is asymptotically almost optimal The formal statement follows.

▶ **Proposition 15** ([13]). *There is no algorithm solving the UNARY BIN PACKING problem in $f(h) \cdot n^{o(h/\log h)}$ time for some function f, where h is the number of bins in the input and n is the input length, unless ETH fails.*

[2] It is known that if ETH is true, then W[1] ≠ FPT [7].

Since our reduction from Unary Bin Packing to ESCAD transforms the parameter linearly and the instance size polynomially, we also have a similar ETH based lower bound parameterized by the vertex cover number for ESCAD.

▶ **Theorem 3.** *There is no algorithm solving ESCAD in $f(k) \cdot n^{o(k/\log k)}$ time for some function f, where k is the vertex cover number of the graph and n is the input length, unless the Exponential Time Hypothesis fails.*

Proof. Follows from the reduction in the proof of Theorem 2 along with Proposition 15. ◀

3.3 NP-hardness of ESCAD on Graphs of Constant Maximum Degree

We show that ESCAD is para-NP-hard when parameterized by the maximum degree.

▶ **Theorem 4.** *ESCAD is* NP*-hard in digraphs where each vertex has* (in, out) *degrees in* $\{(1,6),(6,1)\}$.

Proof. We give a polynomial-time reduction from Vertex Cover on cubic (3-regular) graphs, which is known to be NP-hard [16], to ESCAD. This reduction is a modification of the proof in [16] which shows that Directed Feedback Arc Set is NP-hard. The input to Vertex Cover consists of a graph G and an integer k; the task is to decide whether G has a vertex cover of size at most k. Let (G,k) be an instance of Vertex Cover with n vertices where G is a cubic graph. We construct an ESCAD instance $\mathcal{I}' = (G',k)$ in the following way. The vertex set $V(G') = V(G) \times \{0,1\}$ and the arc set $A(G')$ is defined by the union of the sets $\{((u,0),(u,1)) : u \in V(G)\}$ and $\{((u,1),(v,0))^2 : uv \in E(G)\}$. We call the arcs of the form $((u,0),(u,1))$ *internal arcs* and arcs of the form $((u,1),(v,0))$ *cross arcs*. Towards the correctness of the reduction, we prove the following claim.

▷ Claim 16 (⋆). (G,k) is a yes-instance of Vertex Cover if and only if (G',k) is a yes-instance of ESCAD.

This shows that ESCAD is NP-hard. Moreover, Since G is a cubic graph, every vertex in D' has (in, out) degree equal to $(1,6)$ or $(6,1)$. This completes the proof of Theorem 4. ◀

3.4 Algorithms for ESCAD on Graphs of Bounded Treewidth

Due to Theorem 2, the existence of an FPT algorithm for ESCAD parameterized by various width measures such as treewidth is unlikely. In fact, due to Theorem 3, assuming ETH, even obtaining an algorithm with running time $f(k)n^{o(k/\log k)}$ is not possible, where k is the vertex cover number. On the other hand, this raises a natural algorithmic question – could one obtain an algorithm whose running time matches this lower bound? In this section, we give such an algorithm that is simultaneously, an XP algorithm parameterized by treewidth, an FPT algorithm parameterized by the treewidth and solution size, and also an FPT algorithm parameterized by the treewidth and maximum degree of the input digraph. Moreover, the running time of the algorithm nearly matches the lower bound we have.

Let us note that in the specific case of parameterizing by treewidth and maximum degree, if all we wanted was an FPT algorithm, then we could use Courcelle's theorem at the cost of a suboptimal running time. However, our algorithm in one shot gives us three consequences and as stated earlier, achieves nearly optimal dependence on the treewidth assuming ETH.

Overview of our algorithm. We present a dynamic programming algorithm over tree decompositions. When one attempts to take the standard approach, the main challenge that arises is that by disconnecting strongly connected components, removing an arc can affect vertices far away and hence possibly vertices that have already been forgotten at the current stage of the algorithm. Our solution is to guess the partition of each bag into strongly connected components in the final solution and then keep track of the imbalances of the vertices of the bag under this assumption of components. This allows us to safely forget a vertex as long as its "active" imbalance is zero (any remaining imbalance will be addressed by not strongly connecting the contributing vertices in the future). The remaining difficulty lies in keeping track of how these assumed connections interact with the bag: whether they use vertices already forgotten or those yet to be introduced.

▶ **Theorem 5.** *An ESCAD instance $\mathcal{I} = (G, k)$ can be solved in time $2^{\mathcal{O}(\mathtt{tw}^2)} \cdot (2\alpha+1)^{2\mathtt{tw}} \cdot n^{\mathcal{O}(1)}$ where $\mathtt{tw}$ is the treewidth of deoriented G, Δ is the maximum degree of G, and $\alpha = \min(k, \Delta)$.*

Since the maximum degree is upper bounded by the instance length (recall footnote in Section 1), this gives an XP algorithm parameterized by treewidth alone. However, when in addition to treewidth we parameterize either by the size of the solution or by the maximum degree this gives an FPT algorithm.

▶ **Corollary 17.** *ESCAD is* FPT *parameterized by $\mathtt{tw} + k$,* FPT *parameterized by $\mathtt{tw} + \Delta$, and XP parameterized by $\mathtt{tw}$ alone.*

Recall that in digraphs, multiarcs are permitted. So, we use a variant of the nice tree decomposition notion. This is defined for a digraph G by taking a nice tree decomposition with introduce edge nodes (see Section 2) of the deoriented, simple version of G then expanding each introduce edge node to introduce all parallel copies of arcs one by one. Note that although the new introduce arc nodes introduce *arcs*, the orientation does not affect the decomposition. Let us denote such a tree decomposition of G as $(\mathcal{T}, \{X_t\}_{t \in V(T)})$. Korhonen and Lokshtanov [17] gave a $2^{tw^2} \cdot n^{\mathcal{O}(1)}$-time algorithm that computes an optimal tree decomposition. Moreover, any tree decomposition can be converted to a nice tree decomposition of the same width with introduce edge nodes in polynomial time [5], and the introduce edge nodes can clearly be expanded to introduce arc nodes in polynomial time. Since the running time of our algorithm dominates the time taken for this step, we may assume that we are given such a tree decomposition. Let G_t be the subgraph of the input graph that contains the vertices and arcs introduced in the subtree rooted at t. We refer to G_t as the past and to all other arcs and vertices as the future.

To tackle ESCAD we need to know whether an arc between vertices in a bag is active in the graph minus a hypothetical solution or not. Towards this, we express the reachability of the graph that lies outside (both past and future) of the current bag as follows.

▶ **Definition 18.** *For a set X, let (R, ℓ) be a* reachability arrangement on X *where R is a simple digraph with $V(R) = X$, and ℓ is a labeling $\ell\colon E(R) \to \{\text{direct}, \text{past}, \text{future}\}$.*

Let us use $\ell(u, v)$ to denote $\ell((u, v))$. As reachability arrangement implies which vertices of the bag lie in the same strongly connected components we can determine whether an arc is active by checking that its endpoints lie in the same strongly connected component. We aim to track the balance of the vertices in the bag with respect to all past active arcs.

▶ **Definition 19.** *Given G and R the* active imbalance $b_G^R(v)$ *of a vertex v in G with respect to R is the imbalance of v in the graph H, i.e. $\deg_H^+(v) - \deg_H^-(v)$, where H is the graph induced on G by the vertices of the strongly connected component of R containing v.*

Although the active imbalance is bounded by Δ, it can be large even when the solution is bounded so we want to instead track how much the active imbalance varies between two graphs.

▶ **Definition 20.** *Given G_1, G_2, and R the* offset imbalance *of a vertex v between G_1 and G_2 with respect to R, $\mathrm{off}^R_{G_1,G_2}(v) = b^R_{G_1}(v) - b^R_{G_2}(v)$.*

We will consider the offset imbalance between G_t and $G_t - S$ where S is part of a solution. The following lemma allows us to bound this quantity by the size of the solution.

▶ **Lemma 21** ($\star$)**.** *For each set of arcs $S \subseteq E(G)$, node $t \in V(\mathcal{T})$, simple digraph R on X_t and vertex $v \in X_t$, the offset imbalance of v between $G_t - S$ and G_t with respect to R is between $-|S|$ and $|S|$.*

For a solution S we use a suitable reachability arrangement (R, ℓ), balance labeling B, and part of the solution in the bag W to express a *partial solution*, that is: $S \cap G_t$ along with how vertices of the bag are partitioned into strongly connected components in $G - S$. These give a description of partial solutions that is small enough to guess but detailed enough to admit a dynamic programming approach.

▶ **Definition 22.** *Given a node of the tree decomposition t, a reachability arrangement (R, ℓ) on X_t, a labeling $b \colon V(R) \to [-\alpha, \alpha]$, and a subset of arcs $W \subseteq E(G_t[X_t])$ we call a set of arcs $S \subseteq E(G_t)$* compatible *with R, ℓ, b, W if all of the following parts hold.*

1. *S agrees with W on $G_t[X_t]$, that is $S \cap E(G_t[X_t]) = W$.*
2. *For each arc $e \in \ell^{-1}(\mathrm{direct})$, e is an arc in $G_t[X_t] - S$.*
3. *For each arc $(u, w) \in \ell^{-1}(\mathrm{past})$ there is a path from u to w in $G_t - S$ that contains no vertices from $X_t \setminus \{u, w\}$ (also called path through the past).*
4. *For each arc $(u, w) \in \ell^{-1}(\mathrm{future})$ there is no path through the past from u to w (see part 3) and there is a path from u to w in $G - S$ that contains no vertices from $X_t \setminus \{u, w\}$ (also called path through the future).*
5. *For each vertex $u \in X_t$, the offset imbalance of u between $G_t - S$ and G_t with respect to R is $b(u)$, i.e., $\mathrm{off}^R_{G_t,G_t-S}(u) = b(u)$.*
6. *For each vertex $u \in V(G_t) \setminus X_t$, the active imbalance of u in $G_t - S$ with respect to $(G_t - S) \cup R$ is zero, i.e., $b^{(G_t-S)\cup R}_{G_t-S}(u) = 0$.*

▶ **Observation 23** ($\star$)**.** *Suppose that S is a solution. For all nodes t there exists R, ℓ, b, W such that $S_1 = S \cap E(G_t)$ is compatible with R, ℓ, b, W.*

▶ **Lemma 24** ($\star$)**.** *Suppose that S is a solution and both $S_1 = S \cap E(G_t)$ and $S_2 \subseteq E(G_t)$ are compatible with R, ℓ, b, W. Then $S' = (S \setminus S_1) \cup S_2$ is also a solution.*

The above lemma implies that for fixed t, R, ℓ, b, W all solutions S have the same cardinality of $S \cap G_t$. For fixed t, R, ℓ, b, W to compute existence of some solution S such that $S_1 = S \cap G_t$ is compatible with R, ℓ, b, W, it suffices to compute the minimum cardinality of a subset $S_2 \subseteq E(G_t)$ compatible with R, ℓ, b, W because one can always produce the solution $S' = (S - S_1) \cup S_2$.

Proof of Theorem 5. We will denote by $A[t, R, \ell, b, W]$ the minimum size of an arc subset of G_t that is compatible with R, ℓ, b, and W. In our decomposition $(\mathcal{T}, \{X_t\}_{t \in V(\mathcal{T})})$ the root node r has $X_r = \emptyset$ and $G_r = G$ so $A[r, \emptyset, \emptyset, \emptyset, \emptyset]$ is equal to the minimum size of a solution. In order to compute $A[r, \emptyset, \emptyset, \emptyset, \emptyset]$ we employ the standard bottom up dynamic programming over treewidth decomposition approach.

For leaf nodes $X_t = \emptyset$, hence, the graphs and labelings are also empty and the empty arc set is vacuously compatible with them $A[t, \emptyset, \emptyset, \emptyset, \emptyset] = 0$.

For every non-leaf node t and graph R on X_t we first calculate the strongly connected components of R. Then we can calculate the active imbalance $b^R_{G_t}(v)$ of each vertex $v \in X_t$ in G_t with respect to R. Then for each ℓ, b, and W we calculate $A[t, R, \ell, b, W]$ based on the type of the node t. We give only an informal description here, the full formulae and proofs can be found in the full version.

Introduce vertex node: When t is an introduce vertex node and its child is t' with $X_t = X_{t'} \cup \{v\}$ we know that v will be isolated in G_t so we can discount any reachability arrangements where there are direct or past arcs incident to v. Additionally, the active imbalance on v must be zero. Any new future connections should be reflected in the old reachability arrangement, that is, if the new arrangement contains a future arc from u to v and from v to w there should be a future arc between u and w in the old arrangement. No arcs were introduced or forgotten so the set W remains the same.

Introduce arc node: Assume t introduces arc (u, v) and its child is t'. In any case, if u and v are in different strongly connected components then the new arc is inactive so it does not influence active degrees. We recognize two distinct cases based on whether this new arc belongs to S. On one hand, say the new arc $(u, v) \notin S$, then it may realize a future path from u to v. Also, if u and v are in the same strongly connected component, then the added (u, v) arc changes the active imbalance of u and v by one in G_t but also in $G_t - S$ so the offset imbalance remains the same. On the other hand, if $(u, v) \in S$, then the active degree of its endpoints changes in G_t but it does not change in $G_t - S$, hence, the offset imbalance changes by one. Note that the introduced arc (u, v) may be one among multiple parallel copies of a multiarc – the only minor difference if we did not allow multiarcs would be to not allow the label on (u, v) in t' to be direct.

Forget node: If t is a forget node with child t' such that $X_t = X_{t'} \setminus \{v\}$ then we need to ensure that the forgotten vertex has zero active imbalance in $G_t - S$ and that there are no future arcs incident to it in the old arrangement. Zero active imbalance is equivalent to an offset imbalance of $-b^R_{G_t}(v)$, which we have precalculated. Also, the only change to the remaining reachability arrangement should be new past arcs where there was previously a path through v.

Join node: When merging two nodes t_1 and t_2 to a parent join node t the reachability arrangements should be nearly the same. The notable exception is that past arcs in the parent arrangement can be either past in both child arrangements or we can have past arc in one arrangement while there is a future arc on the other arrangement. In a similar way, we need to consider for each $u \in X_t$ how the imbalance $b(u)$ in G_t is made up of parts in G_{t_1} and G_{t_2}. The new compatible solutions are unions of the solutions compatible with pairs of such arrangements. Their overlap is exactly W so the size of the union is simply the sum of their sizes minus $|W|$.

For a fixed node t there are $4^{\mathtt{tw}^2}$ reachability arrangements on X_t, $(2\alpha + 1)^{\mathtt{tw}}$ possible b's, and $2^{\mathtt{tw}^2}$ possible W's. Both introduce vertex and introduce arc node compute their entry from a fixed entry of their child node in $n^{\mathcal{O}(1)}$ time. Forget node is computed in $2^{\mathtt{tw}} \cdot 4^{\mathtt{tw}} \cdot n^{\mathcal{O}(1)}$ while join node is computed in $3^{\mathtt{tw}^2} \cdot (2\alpha + 1)^{\mathtt{tw}} \cdot n^{\mathcal{O}(1)}$ time.

It is known that the total number of nodes in the nice tree decomposition with introduce arc nodes is $n^{\mathcal{O}(1)}$ and it can be observed that this still holds for the extension on multiarcs. Hence, the overall run time is

$$\left(4^{\mathtt{tw}^2} \cdot (2\alpha+1)^{\mathtt{tw}} \cdot 2^{\mathtt{tw}^2}\right) \cdot \left(2^{\mathtt{tw}} \cdot 4^{\mathtt{tw}} + 3^{\mathtt{tw}^2} \cdot (2\alpha+1)^{\mathtt{tw}}\right) \cdot n^{\mathcal{O}(1)} = 24^{\mathtt{tw}^2} \cdot (2\alpha+1)^{2\mathtt{tw}} \cdot n^{\mathcal{O}(1)}. \blacktriangleleft$$

4 Our Results for ESCAD on Simple Digraphs

In this section, we study ESCAD on simple digraphs, which we formally define as follows.

SIMPLE EULERIAN STRONG COMPONENT ARC DELETION (SESCAD)
Input: A simple digraph G, an integer k
Question: Is there a subset $R \subseteq E(G)$ of size $|R| \leq k$ such that in $G - R$ each strongly connected component is Eulerian?

Let us begin by stating a simple observation that enables us to make various inferences regarding the complexity of SESCAD based on the results we have proved for ESCAD.

▶ **Observation 25.** *Consider an ESCAD instance $\mathcal{I} = (G, k)$. If we subdivide every arc (u, v) into $(u, w), (w, v)$ (using a new vertex w) then we get an equivalent SESCAD instance $\mathcal{I}' = (G', k)$ with $|V(G')| = |V(G)| + |E(G)|$ and $|E(G')| = 2|E(G)|$. Moreover, each arc of the solution to $\mathcal{I}$ is mapped to one respective arc of the subdivision and vice versa.*

4.1 Hardness Results for SESCAD

We first discuss the implications of Theorem 1, Theorem 2 and Theorem 4 for SESCAD along with Observation 25.

▶ **Corollary 26.** *SESCAD is* W[1]*-hard when parameterized by the solution size.*

Proof. Follows from Theorem 1 and Observation 25. ◀

▶ **Observation 27.** *If we subdivide all arcs in a digraph G that has a vertex cover X, we get a simple digraph G' such that $G' - X$ is the disjoint union of directed stars.*

▶ **Corollary 28.** *SESCAD is* W[1]*-hard parameterized by minimum modulator size to disjoint union of directed stars.*

Using the stronger assumption of ETH, we have the following result.

▶ **Theorem 29.** *There is no algorithm solving SESCAD in $f(k) \cdot n^{o(k/\log k)}$ time for some function f, where k is the size of the smallest vertex set that must be deleted from the input graph to obtain a disjoint union of directed stars and n is the input length, unless the Exponential Time Hypothesis fails.*

Proof. The reduction in the proof of Theorem 2 along with Proposition 15, Observation 25 and Observation 27 implies the statement. ◀

Note that the above result rules out an FPT algorithm for SESCAD parameterized by various width measures such as treewidth and even treedepth.

▶ **Theorem 30.** *SESCAD is* NP*-hard in simple digraphs where each vertex has* (in, out) *degrees in* $\{(1, 1), (1, 6), (6, 1)\}$.

Proof. Follows from Theorem 4 and Observation 25. ◀

4.2 FPT Algorithms for SESCAD

Firstly, the FPT algorithms discussed in the previous section naturally extend to SESCAD. However, for SESCAD, the lower bound parameterized by modulator to a disjoint union of directed stars leaves open the question of parameterizing by larger parameters. For instance, the vertex cover number.

To address this gap, we provide an FPT algorithm for SESCAD parameterized by *vertex integrity*, a parameter introduced by Barefoot et al. [1].

▶ **Definition 31** (Vertex Integrity). An undirected graph $G = (V, E)$ has *vertex integrity k* if there exists a set of vertices $M \subseteq V$, called a *k-separator*, of size at most k such that when removed each connected component has size at most $k - |M|$. A directed graph has vertex integrity k if and only if the underlying undirected graph has vertex integrity k. The notion of a k-separator in digraphs carries over naturally from the undirected setting.

FPT algorithms parameterized by vertex integrity have gained popularity in recent years due to the fact that several problems known to be W[1]-hard parameterized even by treedepth can be shown to be FPT when parameterized by the vertex integrity [10]. Since Corollary 28 rules out FPT algorithms for SESCAD parameterized by treedepth, it is natural to explore SESCAD parameterized by vertex integrity and our positive result thus adds SESCAD to the extensive list of problems displaying this behavior.

Moreover, this FPT algorithm parameterized by vertex integrity implies that SESCAD is also FPT when parameterized by the vertex cover number and shows that our reduction for ESCAD parameterized by the vertex cover number requires multiarcs for fundamental reasons and cannot be just adapted to simple digraphs with more work.

We will use as a subroutine the well-known FPT algorithm for ILP-Feasibility. The ILP-Feasibility problem is defined as follows. The input is a matrix $A \in \mathbb{Z}^{m \times p}$ and a vector $b \in \mathbb{Z}^{m \times 1}$ and the objective is to find a vector $\bar{x} \in \mathbb{Z}^{p \times 1}$ satisfying the m inequalities given by A, that is, $A \cdot \bar{x} \leq b$, or decide that such a vector does not exist.

▶ **Proposition 32** ([14, 15, 9]). *ILP-Feasibility can be solved using $\mathcal{O}(k^{2.5k+o(k)} \cdot L)$ arithmetic operations and space polynomial in L, where L is the number of bits in the input and k is the number of variables.*

▶ **Theorem 6.** *ESCAD on simple digraphs is* FPT *parameterized by the vertex integrity of the graph.*

Proof. Consider an instance (G, p) of SESCAD, where G has vertex integrity at most k. Suppose that this is a yes-instance with a solution S and let M be a k-separator of G. Without loss of generality, assume that $V(G) = [n]$ and $M = [|M|]$. In our algorithm, we only require the fact that since M is a k-separator in a digraph G, every weakly connected component of $G - M$ has size at most k (recall, the definition of vertex integrity bounds the component sizes even more). Further, we remark that our algorithm does not require a k-separator to be given as input since there is an FPT algorithm parameterized by k to compute it [8].

We next guess those arcs of S that have both endpoints in M, remove them and adjust p accordingly. The number of possible guesses is $2^{\mathcal{O}(k^2)}$. Henceforth, we assume that every arc in the hypothetical solution S has at least one endpoint disjoint from M.

We next guess the reachability relations between the vertices of M in $G - S$. The correct guess is called the *reachability signature* of M in $G - S$, denoted by σ, which is a set of ordered pairs where, for every $m_1, m_2 \in M$, $(m_1, m_2) \in \sigma$ if and only if m_2 is reachable from m_1 in $G - S$. The number of possibilities for σ is clearly bounded by $2^{\mathcal{O}(k^2)}$.

For every simple digraph comprised of at most $|M|+k$ vertices and every possible injective mapping λ of M to the vertices of this digraph, we define the *type* of this digraph as the label-preserving isomorphism class with the labeling λ. Denote the set of all types by Types. For each type $\tau \in \mathsf{Types}$, we denote by G_τ a fixed graph of this type that we can compute in time depending only on k. Due to the labeling injectively mapping M to the vertices of G_τ, we may assume that $M \subseteq V(G_\tau)$.

The number of types is clearly bounded by a function of k and for each weakly connected component (from now onwards, simply called a component) C of $G - M$ and graph $G_C = G[C \cup M]$ with the vertices of M mapped to themselves by the identity labeling on M, denoted λ_M, we compute the type of the graph G_C. From now on, we drop the explicit reference to λ_M as it will be implied whenever we are handling the graph G_C. For every type τ, we also compute the number n_τ of components C such that G_C is of type τ. Since the type of each G_C can be computed in $f(k)$-time for some function f, this step takes FPT time.

Following that, for every component C, and every arc set S_C in G_C, we check whether the type of $G_C - S_C$ (with labeling λ_M) is *compatible* with σ. To be precise, for a set S_C of arcs in the digraph G_C, we verify that every vertex of C is balanced in its strongly connected component in the graph $G'_C = G_C - S_C + \sigma$. If the answer to this check is yes, then this is a compatible type. Notice that by adding the ordered pairs in σ as arcs to $G_C - S_C$, we ensure that the arcs of the graph we take into account in this check on balances of the vertices in C (i.e., active arcs) are exactly all those arcs that are already in strong$(G_C - S_C)$ plus those arcs of $G_C - S_C$ that *would* be inside a strongly connected component *if* the relations in σ were realized. Since each component C has size bounded by k, the number of possibilities for S_C is bounded by a function of k for each component (here, we crucially use the fact that we have a simple digraph), and hence, in FPT time, we can compute a table Γ stating, for every C and S_C subset of arcs in G_C, whether the type of $G_C - S_C$ is compatible with σ.

Notice that for each component, deleting the arcs of the hypothetical solution S from each component C transitions G_C from one type to another type that is compatible with σ. To be precise, for each C and set $S_C = S \cap A(G_C)$, we can think of S_C as taking G_C from the type of G_C (call it τ_1) to the type of $G_C - S_C$ (call it τ_2), at cost $|S_C|$. Moreover, the type τ_2 is compatible with σ. Thus, the table Γ encodes the cost of transitioning each graph G_C to a type compatible with σ. This can be expressed by a value $\text{cost}(\tau_1, \tau_2)$ for every pair of types. If τ_2 is not compatible with σ, then set this value to be prohibitively high, say the number of arcs in G plus one. Otherwise, $\text{cost}(\tau_1, \tau_2)$ is given by the table Γ.

In our next step, we guess a set of $O(k^2)$ types such that for every pair of vertices $m_1, m_2 \in M$, if σ requires that m_1 can reach m_2, then there is a sequence of vertices of M starting at m_1 and ending in m_2 such that for every consecutive ordered pair (x, y) in this sequence, either (x, y) is an arc in $G[M]$ (and since it is not already deleted, it is disjoint from S) or there is an x-y path with all internal vertices through a subgraph that belongs to one of these $\mathcal{O}(k^2)$ types. Call this set of types T^*. The bound on the size of T^* comes from the fact that there are $\mathcal{O}(k^2)$ pairs in σ.

Finally, whether or not the vertices of M are balanced in *strong*$(G - S)$ is determined entirely by the number of graphs of each type in $G - S$ subject to the types in T^* occurring. So, for every type, we determine the imbalance imposed by the type on each vertex of M (taking σ into account). To be precise, for every type τ and vertex $u \in M$, the imbalance on u due to τ is denoted by $I(\tau, u)$ and is obtained by subtracting the number of active incoming arcs on u from the number of active outgoing arcs on u, where an arc $(p, q) \in A(G_\tau)$ where $u \in p, q$ is active, if and only if it lies in the same strongly connected component as u in the graph $G_\tau + \sigma$.

All of the above requirements can be formulated as an ILP-Feasibility instance with $f(k)$ variables that effectively minimizes the total costs of all the required type transitions. More precisely, for every pair of types τ_1 and τ_2, we have a variable x_{τ_1,τ_2} that is intended to express the number of graphs G_C of type τ_1 that transition to type τ_2. We only need to consider variables x_{τ_1,τ_2} where τ_1 is the type of some G_C and τ_2 is compatible with σ. So, we restrict our variable set to this. Moreover, for every τ that is compatible with σ, we have a variable y_τ that is intended to express the number of components C such that G_C transitions to type τ.

Then, we have constraints that express the following:

1. The cost of all the type transitions is at most p.

$$\sum_{\tau_1,\tau_2 \in \mathsf{Types}} cost(\tau_1, \tau_2) \cdot x_{\tau_1,\tau_2} \le p$$

2. For each type τ in T^*, there is at least one transition to τ. This will ensure that the reachability relations required by σ are achieved.

$$\sum_{\tau_1 \in \mathsf{Types}} x_{\tau_1,\tau} \ge 1$$

3. For every component C, G_C transitions to some type compatible with σ. So, for every type τ, we have:

$$\sum_{\tau_2 \in \mathsf{Types}} x_{\tau,\tau_2} = n_\tau$$

   Recall that n_τ denotes the number of components C such that G_C is of type τ and we have computed it already.

4. The number of components C such that G_C transitions to type τ, is given by summing up the values of $x_{\tau_1,\tau}$ over all possible values of τ_1.

$$\sum_{\tau_1 \in \mathsf{Types}} x_{\tau_1,\tau} = y_\tau$$

5. The total imbalance imposed on each vertex of M by the existing arcs incident to it, plus the imbalance imposed on it by the types to which we transition, adds up to 0.
   For each $u \in M$, let ρ_u denote the imbalance on u imposed by those arcs of $G[M]$ that are incident to u and active in the graph $G[M] + \sigma$. The imbalance imposed on u by a particular type τ is $I(\tau, u)$ and this needs to be multiplied by the number of "occurrences" of this type after removing the solution, i.e., the value of y_τ.
   Hence, we have the following constraint for every $u \in M$.

$$\rho_u + \sum_{\tau \in \mathsf{Types}} I(\tau, u) y_\tau = 0$$

6. Finally, we need the variables to all get non-negative values. So, for every $\tau_1, \tau_2 \in \mathsf{Types}$, we add $x_{\tau_1,\tau_2} \ge 0$ and for every $\tau \in \mathsf{Types}$, $y_\tau \ge 0$.

It is straightforward to convert the above constraints into the form of an instance of ILP-Feasibility. Since the number of variables is a function of k, Proposition 32 can be used to decide feasibility in FPT time. From a solution to the ILP-Feasibility instance, it is also straightforward to recover a solution to our instance by using the table Γ. ◀

5 Conclusions

We have resolved the open problem of Cechlárová and Schlotter [3] on the parameterized complexity of the Eulerian Strong Component Arc Deletion problem by showing that it is W[1]-hard and accompanied it with further hardness results parameterized by the vertex cover number and max-degree of the graph. On the positive side, we showed that though the problem is inherently difficult in general, certain combined parameterizations (such as treewidth plus either max-degree or solution size) offer a way to obtain FPT algorithms.

Our work points to several natural future directions of research on this problem.

1. Design of (FPT) approximation algorithms for ESCAD?
2. ESCAD parameterized by the solution size is FPT on tournaments [4]. For which other graph classes is the problem FPT by the same parameter?
3. Our FPT algorithm for SESCAD parameterized by vertex integrity is only aimed at being a characterization result and we have not attempted to optimize the parameter dependence. So, a natural follow up question is to obtain an algorithm that is as close to optimal as possible.
4. Are there parameterizations upper bounding the solution size, for which ESCAD is FPT? For instance, the size of the minimum directed feedback arc set of the input digraph. Notice that in the reduction of Theorem 1, we obtain instances with unboundedly large minimum directed feedback arc sets due to the imbalance gadgets starting at the vertex s_j for some color class j and ending at the vertices in $\{x_u \mid u \in \text{color class } j\}$.

References

1 C. A. Barefoot, R. Entringer, and H. C. Swart. Vulnerability in graphs—a comparative survey. *JCMCC*, 1:13–22, 1987.

2 Václav Blazej, Satyabrata Jana, M. S. Ramanujan, and Peter Strulo. On the parameterized complexity of eulerian strong component arc deletion. *CoRR*, abs/2408.13819, 2024. doi:10.48550/arXiv.2408.13819.

3 Katarína Cechlárová and Ildikó Schlotter. Computing the deficiency of housing markets with duplicate houses. In Venkatesh Raman and Saket Saurabh, editors, *Parameterized and Exact Computation - 5th International Symposium, IPEC 2010, Chennai, India, December 13-15, 2010. Proceedings*, volume 6478 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 2010. doi:10.1007/978-3-642-17493-3_9.

4 Robert Crowston, Gregory Z. Gutin, Mark Jones, and Anders Yeo. Parameterized eulerian strong component arc deletion problem on tournaments. *Inf. Process. Lett.*, 112(6):249–251, 2012. doi:10.1016/J.IPL.2011.11.014.

5 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

6 Marek Cygan, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Ildikó Schlotter. Parameterized complexity of eulerian deletion problems. *Algorithmica*, 68(1):41–61, 2014. doi:10.1007/S00453-012-9667-X.

7 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

8 Pål Grønås Drange, Markus S. Dregi, and Pim van 't Hof. On the computational complexity of vertex integrity and component order connectivity. *Algorithmica*, 76(4):1181–1202, 2016. doi:10.1007/S00453-016-0127-X.

9 András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Comb.*, 7(1):49–65, 1987. doi:10.1007/BF02579200.

10 Tatsuya Gima, Tesshu Hanaka, Masashi Kiyomi, Yasuaki Kobayashi, and Yota Otachi. Exploring the gap between treedepth and vertex cover through vertex integrity. *Theor. Comput. Sci.*, 918:60–76, 2022. doi:10.1016/J.TCS.2022.03.021.

11 Alexander Göke, Dániel Marx, and Matthias Mnich. Parameterized algorithms for generalizations of directed feedback vertex set. *Discret. Optim.*, 46:100740, 2022. doi:10.1016/J.DISOPT.2022.100740.

12 Prachi Goyal, Pranabendu Misra, Fahad Panolan, Geevarghese Philip, and Saket Saurabh. Finding even subgraphs even faster. *J. Comput. Syst. Sci.*, 97:1–13, 2018. doi:10.1016/J.JCSS.2018.03.001.

13 Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *J. Comput. Syst. Sci.*, 79(1):39–49, 2013. doi:10.1016/J.JCSS.2012.04.004.

14 Hendrik W. Lenstra Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983. doi:10.1287/MOOR.8.4.538.

15 Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987. doi:10.1287/MOOR.12.3.415.

16 Richard M. Karp. Reducibility among combinatorial problems. In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, pages 219–241. Springer, 2010. doi:10.1007/978-3-540-68279-0_8.

17 Tuukka Korhonen and Daniel Lokshtanov. An improved parameterized algorithm for treewidth. In *STOC*, pages 528–541. ACM, 2023. doi:10.1145/3564246.3585245.