Algorithmica (2021) 83:1605–1637
https://doi.org/10.1007/s00453-020-00795-3

On Structural Parameterizations of the Edge Disjoint Paths Problem

Robert Ganian[1] · **Sebastian Ordyniak**[2] · **M. S. Ramanujan**[3]

Received: 30 March 2020 / Accepted: 22 December 2020 / Published online: 25 January 2021

Abstract
In this paper we revisit the classical edge disjoint paths (EDP) problem, where one is given an undirected graph G and a set of terminal pairs P and asks whether G contains a set of pairwise edge-disjoint paths connecting every terminal pair in P. Our focus lies on structural parameterizations for the problem that allow for efficient (polynomial-time or FPT) algorithms. As our first result, we answer an open question stated in Fleszar et al. (Proceedings of the ESA, 2016), by showing that the problem can be solved in polynomial time if the input graph has a feedback vertex set of size one. We also show that EDP parameterized by the treewidth and the maximum degree of the input graph is fixed-parameter tractable. Having developed two novel algorithms for EDP using structural restrictions on the input graph, we then turn our attention towards the augmented graph, i.e., the graph obtained from the input graph after adding one edge between every terminal pair. In constrast to the input graph, where EDP is known to remain NP-hard even for treewidth two, a result by Zhou et al. (Algorithmica 26(1):3--30, 2000) shows that EDP can be solved in non-uniform polynomial time if the augmented graph has constant treewidth; we note that the possible improvement of this result to an FPT-algorithm has remained open since then. We show that this is highly unlikely by establishing the W[1]-hardness of the problem parameterized by the treewidth (and even feedback vertex set) of the augmented graph. Finally, we develop an FPT-algorithm for EDP by exploiting a novel structural parameter of the augmented graph.

Keywords Edge disjoint path problem · Feedback vertex set · Treewidth · Fracture number · Parameterized complexity

A preliminary and shortened version of this paper has been accepted at ISAAC 2017.

✉ Sebastian Ordyniak
sordyniak@gmail.com

[1] Algorithms and Complexity Group, TU Wien, Vienna, Austria

[2] School of Computing, University of Leeds, Leeds, UK

[3] University of Warwick, Coventry, UK

1 Introduction

The Edge Disjoint Paths (EDP) and Node Disjoint Paths (NDP) are fundamental routing graph problems. In the EDP (NDP) problem the input is a graph G, and a set P containing k pairs of vertices and the objective is to decide whether there is a set of k pairwise edge disjoint (respectively vertex disjoint) paths connecting each pair in P. These problems and their optimization versions–MaxEDP and MaxNDP–have been at the center of numerous results in structural graph theory, approximation algorithms, and parameterized algorithms [4, 10, 12, 17, 21, 23, 26, 27, 29].

When k is a part of the input, both EDP and NDP are known to be NP-complete [20]. Robertson and Seymour's seminal work in the Graph Minors project [27] provides an $\mathcal{O}(n^3)$ time algorithm for both problems for every fixed value of k. In the realm of *Parameterized Complexity*, their result can be interpreted as *fixed-parameter* algorithms for EDP and NDP parameterized by k. Here, one considers problems associated with a certain numerical parameter k and the central question is whether the problem can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ where f is a computable function and n the input size; algorithms with running time of this form are called FPT-algorithms [5, 7, 13].

While the aforementioned research considered the number of paths to be the parameter, another line of research investigates the effect of *structural parameters* of the input graphs on the complexity of these problems. Fleszar, Mnich, and Spoerhase [12] initiated the study of NDP and EDP parameterized by the feedback vertex set number (the size of the smallest feedback vertex set) of the input graph and showed that EDP remains NP-hard even on graphs with feedback vertex set number two. Since EDP is known to be polynomial time solvable on forests [17], this left only the case of feedback vertex set number one open, which they conjectured to be polynomial time solvable. Our first result is a positive resolution of their conjecture.

Theorem 1 EDP *can be solved in time* $\mathcal{O}(|P||V(G)|^{\frac{5}{2}})$ *on graphs with feedback vertex set number one.*

A key observation behind the polynomial-time algorithm is that an EDP instance with a feedback vertex set $\{x\}$ is a yes-instance if and only if, for every tree T of $G - \{x\}$, it is possible to connect all terminal pairs in T either to each other or to x through pairwise edge disjoint paths in T.The main ingredient of the algorithm is then a dynamic programming procedure that determines whether this is indeed possible for a tree T of $G - \{x\}$.

Continuing to explore structural parameterizations for the input graph of an EDP instance, we then show that even though EDP is NP-complete when the *input graph* has treewidth two, it becomes fixed-parameter tractable if we additionally parameterize by the maximum degree. There are, in fact, two ways one may attempt to prove this: via a reduction to NDP (see, e.g., Proposition 1 in a follow-up paper [16]), or via an explicit dynamic programming algorithm. Here, we use the latter approach, which has the advantage of yielding a better running time (see Sect. 4).

Theorem 2 EDP *is fixed-parameter tractable parameterized by the treewidth and the maximum degree of the input graph*.

Having explored the algorithmic applications of structural restrictions on the input graph for EDP, we then turn our attention towards similar restrictions on the *augmented graph* of an EDP instance (G, P), i.e., the graph obtained from G after adding an edge between every pair of terminals in P. Whereas EDP is NP-complete even if the input graph has treewidth at most two [26], it can be solved in non-uniform polynomial time if the treewidth of the augmented graph is bounded [29]. It has remained open whether EDP is fixed-parameter tractable parameterized by the treewidth of the augmented graph; interestingly, this has turned out to be the case for the strongly related multicut problems [18]. Surprisingly, we show that this is not the case for EDP, by establishing the W[1]-hardness of the problem parameterized by not only the treewidth but also by the feedback vertex set number of the augmented graph.

Theorem 3 EDP *is* w *[1]-hard parameterized by the feedback vertex set number of the augmented graph*..

Motivated by this strong negative result, our next aim was to find natural structural parameterizations for the augmented graph of an EDP instance for which the problem becomes fixed-parameter tractable. Towards this aim, we introduce the *fracture number*, which informally corresponds to the size of a minimum vertex set S such that the size of every component in the graph minus S is small (has size at most $|S|$). We show that EDP is fixed-parameter tractable parameterized by this new parameter.

Theorem 4 EDP *is fixed-parameter tractable parameterized by the fracture number of the augmented graph*.

We note that the reduction in [12, Theorem 6] excludes the applicability of the fracture number of the *input graph* by showing that EDP is NP-complete even for instances with fracture number at most three. Finally, we complement Theorem 4 by showing that bounding the number of terminal pairs in each component instead of the its size is not sufficient to obtain fixed-parameter tractability. Indeed, we show that EDP is NP-hard even on instances (G, P) where the augmented graph G^P has a deletion set D of size 6 such that every component of $G^P \setminus D$ contains at most 1 terminal pair. We note that a parameter similar to the fracture number has recently been used to obtain FPT-algorithms for Integer Linear Programming [9].

2 Preliminaries

2.1 Basic Notation

We use standard terminology for graph theory, see for instance [6]. Given a graph G, we let $V(G)$ denote its vertex set, $E(G)$ its edge set and by $V(E')$ the set of vertices

incident with the edges in E', where $E' \subseteq E(G)$. The (open) neighborhood of a vertex $x \in V(G)$ is the set $\{y \in V(G) : xy \in E(G)\}$ and is denoted by $N_G(x)$. For a vertex subset X, the neighborhood of X is defined as $\bigcup_{x \in X} N_G(x) \setminus X$ and denoted by $N_G(X)$. For a vertex set A, we use $G - A$ to denote the graph obtained from G by deleting all vertices in A, and we use $G[A]$ to denote the *subgraph induced on* A, i.e., $G - (V(G) \setminus A)$. A *forest* is a graph without cycles, and a vertex set X is a *feedback vertex set* (*FVS*) if $G - X$ is a forest. We use $[i]$ to denote the set $\{0, 1, \ldots, i\}$. The *feedback vertex set number* of a graph G, denoted by $\mathbf{fvs}(G)$, is the smallest integer k such that G has a feedback vertex set of size k. We denote by $\overline{G}$ the underlying undirected graph of a directed graph G.

2.2 Edge Disjoint Path Problem

Throughout the paper we consider the following problem.

Edge Disjoint Paths (EDP)	
Input:	A graph G a set P of terminal pairs, i.e., a set of subsets of $V(G)$ of size two.
Question:	Is there a set of pairwise edge disjoint paths connecting every set of terminal pairs in P?

Let (G, P) be an instance of EDP; for brevity, we will sometimes denote a terminal pair $\{s, t\} \in P$ simply as st. For a subgraph H of G, we denote by $P(H)$ the subset of P containing all sets that have a non-empty intersection with $V(H)$ and for $P' \subseteq P$, we denote by $\widetilde{P'}$ the set $\bigcup_{p \in P'} p$.

To streamline our presentation, we will assume that each vertex $v \in V(G)$ occurs in at most one terminal pair, each vertex in a terminal pair has degree 1 in G, and each terminal pair is not adjacent to each other. These assumptions have also been used in previous works on EDP (such as in the work of Fleszar et al. [12]), and can be ensured by a simple reduction: we can add a new leaf vertex for terminal, attach it to the original terminal, and replace the original terminal with the leaf vertex [29]. However, we do remark that the reduction may in fact increase the value of some of our parameterizations; in Sect. 6 we showcase how this issue can be circumvented for one of our parameters, notably the *fracture number*.

Definition 1 [29] The *augmented graph* of (G, P) is the graph G^P obtained from G by adding edges between each terminal pair, i.e.,
$G^P = (V(G), E(G) \cup P)$.

2.3 Parameterized Complexity

A *parameterized problem* $\mathcal{P}$ is a subset of $\Sigma^* \times \mathbb{N}$ for some finite alphabet Σ. Let $L \subseteq \Sigma^*$ be a classical decision problem for a finite alphabet, and let p be a

non-negative integer-valued function defined on Σ^*. Then L *parameterized by* p denotes the parameterized problem $\{ (x, p(x)) \mid x \in L \}$ where $x \in \Sigma^*$. For a problem instance $(x, k) \in \Sigma^* \times \mathbb{N}$ we call x the main part and k the parameter. A parameterized problem $\mathcal{P}$ is *fixed-parameter tractable* (FPT in short) if a given instance (x, k) can be solved in time $\mathcal{O}(f(k) \cdot p(|x|))$ where f is an arbitrary computable function of k and p is a polynomial function; we call algorithms running in this time *FPT-algorithms*.

Parameterized complexity classes are defined with respect to *FPT-reducibility*. A parameterized problem $\mathcal{P}$ is *FPT-reducible* to $\mathcal{Q}$ if in time $f(k) \cdot |x|^{O(1)}$, one can transform an instance (x, k) of $\mathcal{P}$ into an instance (x', k') of $\mathcal{Q}$ such that $(x, k) \in \mathcal{P}$ if and only if $(x', k') \in \mathcal{Q}$, and $k' \leq g(k)$, where f and g are computable functions depending only on k. Owing to the definition, if $\mathcal{P}$ FPT-reduces to $\mathcal{Q}$ and $\mathcal{Q}$ is fixed-parameter tractable then P is fixed-parameter tractable as well. Central to parameterized complexity is the following hierarchy of complexity classes, defined by the closure of canonical problems under FPT-reductions:

$$\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{XP}.$$

All inclusions are believed to be strict. In particular, $\mathsf{FPT} \neq \mathsf{W}[1]$ under the Exponential Time Hypothesis.

The class $\mathsf{W}[1]$ is the analog of NP in parameterized complexity. A major goal in parameterized complexity is to distinguish between parameterized problems which are in FPT and those which are $\mathsf{W}[1]$-hard, i.e., those to which every problem in $\mathsf{W}[1]$ is FPT-reducible. There are many problems shown to be complete for $\mathsf{W}[1]$, or equivalently $\mathsf{W}[1]$-complete, including the Multi-Colored Clique (MCC) problem [7]. We refer the reader to the respective monographs [5, 7, 13] for an in-depth introduction to parameterized complexity.

2.4 Integer Linear Programming

Our algorithms use an Integer Linear Programming (ILP) subroutine. ILP is a well-known framework for formulating problems and a powerful tool for the development of FPT-algorithms for optimization problems.

Definition 2 (*p-Variable Integer Linear Programming Optimization*) Let $A \in \mathbb{Z}^{q \times p}$, $b \in \mathbb{Z}^{q \times 1}$ and $c \in \mathbb{Z}^{1 \times p}$. The task is to find a vector $\bar{x} \in \mathbb{Z}^{p \times 1}$ which minimizes the objective function $c\bar{x}$ and satisfies all q inequalities given by A and b, specifically satisfies $A \cdot \bar{x} \geq b$. The number of variables p is the parameter.

Lenstra [24] showed that p -ILP, together with its optimization variant p -OPT-ILP (defined above), are in FPT. His running time was subsequently improved by Kannan [19] and Frank and Tardos [14] (see also [11]).

Proposition 1 [11, 14, 19, 24] *p -OPT-ILP can be solved using $\mathcal{O}(p^{2.5p+o(p)} \cdot L)$ arithmetic operations in space polynomial in L, where L is the number of bits in the input.*

2.5 Treewidth

A *tree-decomposition* of a graph $G = (V, E)$ is a pair $(T, \{B_t : t \in V(T)\})$ where $B_t \subseteq V$ for every $t \in V(T)$ and T is a tree such that:

(1) for each edge $uv \in E$, there is a $t \in V(T)$ such that $\{u, v\} \subseteq B_t$, and
(2) for each vertex $v \in V$, $T[\{ t \in V(T) \mid v \in B_t \}]$ is a non-empty (connected) tree.

The *width* of a tree-decomposition is $\max_{t \in V(T)} |B_t| - 1$. The *treewidth* [22] of G is the minimum width taken over all tree-decompositions of G and it is denoted by $\mathbf{tw}(G)$. We call the elements of $V(T)$ *nodes* and B_t *bags*.

While it is possible to compute the treewidth exactly using an FPT-algorithm [1], the asymptotically best running time is achieved by using the recent state-of-the-art 5-approximation algorithm of Bodlaender et al. [2].

Fact 1 [2] There exists an algorithm which, given an n-vertex graph G and an integer k, in time $2^{\mathcal{O}(k)} \cdot n$ either outputs a tree-decomposition of G of width at most $5k + 4$ and $\mathcal{O}(n)$ nodes, or correctly determines that $\mathbf{tw}(G) > k$.

It is well known that, for every clique over $Z \subseteq V(G)$ in G, it holds that every tree-decomposition of G contains an element B_t such that $Z \subseteq B_t$ [22]. Furthermore, if t' separates a node t from another node t'' in T, then $B_{t'}$ separates $B_t \setminus B_{t'}$ from $B_{t''} \setminus B_{t'}$ in G [22]; this *inseparability property* will be useful in some of our later proofs. A tree-decomposition $(T, B_t : t \in V(T))$ of a graph G is *nice* if the following conditions hold:

1. T is rooted at a node r such that $|B_r| = \emptyset$.
2. Every node of T has at most two children.
3. If a node t of T has two children t_1 and t_2, then $B_t = B_{t_1} = B_{t_2}$; in that case we call t a *join node*.
4. If a node t of T has exactly one child t', then exactly one of the following holds:
   (a) $|B_t| = |B_{t'}| + 1$ and $B_{t'} \subset B_t$; in that case we call t an *introduce node*.
   (b) $|B_t| = |B_{t'}| - 1$ and $B_t \subset B_{t'}$; in that case we call t a *forget node*.
5. If a node t of T is a leaf, then $|B_t| = 1$; we call these *leaf nodes*.

The main advantage of nice tree-decompositions is that they allow the design of much more transparent dynamic programming algorithms, since one only needs to deal with four specific types of nodes. It is well known (and easy to see) that for every fixed k, given a tree-decomposition of a graph $G = (V, E)$ of width at most k and with $\mathcal{O}(|V|)$ nodes, one can construct in linear time a nice tree-decomposition of G with $\mathcal{O}(|V|)$ nodes and width at most k [3]. Given a node t in T, we let Y_t be the set of all vertices contained in the bags of the subtree rooted at t, i.e., $Y_t = B_t \cup \bigcup_{p \text{ is separated from the root by t}} B_p$.

3 Closing the Gap on Graphs of Feedback Vertex Number One

In this section we develop a polynomial-time algorithm for EDP restricted to graphs with feedback vertex set number one. We refer to this particular variant as Simple Edge Disjoint Paths (SEDP): given an EDP instance (G, P) and a FVS $X = \{x\}$, solve (G, P).

Simple Edge Disjoint Paths (SEDP)	
Input:	A graph G, an FVS $X = \{x\} \subseteq V(G)$ for G and a set P of terminal pairs, i.e., a set of subsets of $V(G)$ of size two.
Question:	Is there a set of pairwise edge disjoint paths connecting every set of two terminals in P?

Additionally to our standard assumptions about EDP (given in Sect. 2.2), we will assume that: (1) every neighbor of x in G is a leaf in $G - X$, (2) x is not a terminal, i.e., $x \notin \widetilde{P}$, and (3) every tree T in $G - X$ is rooted in a vertex r that is not a terminal. Property (1) can be ensured by an additional leaf vertex l to any non-leaf neighbor n of x, removing the edge $\{n, x\}$ and adding the edge $\{l, x\}$ to G. Property (2) can be ensured by adding an additional leaf vertex l to x and replacing x with l in P and finally (3) can be ensured by adding a leaf vertex l to any non-terminal vertex r in T and replacing r with l in P.

A key observation behind our algorithm for SEDP is that whether or not an instance $\mathcal{I} = (G, P, X)$ has a solution merely depends on the existence of certain sets of pairwise edge disjoint paths in the trees T in $G - X$. In particular, as we will show in Lemma 1 later on, $\mathcal{I}$ has a solution if and only if every tree T in $G - X$ is $\gamma_\emptyset$-connected (see Definition 3). The main ingredient of the algorithm is then a bottom-up dynamic programming algorithm that determines whether a tree T in $G - X$ is $\gamma_\emptyset$-connected. We now define the various connectivity states of subtrees of T that we need to keep track of in the dynamic programming table.

Definition 3 Let T be a tree in $G - X$ rooted at r (recall that we can assume that r is not in $\widetilde{P}$), $t \in V(T)$, let S be a set of pairwise edge disjoint paths in $G[T_t \cup X]$, and let $P' \subseteq P(T_t)$, where T_t is the subtree of T rooted at t.

We say that the set S *$\gamma_\emptyset$-connects* P' in $G[T_t \cup X]$ if for every $a \in \widetilde{P'} \cap V(T_t)$, the set S either contains an a-x path disjoint from b, or it contains an a-b path disjoint from x, where $\{a, b\} \in P'$. Moreover, for $\ell \in \{\gamma_X\} \cup P(T_t)$, we say that the set S *ℓ-connects* T_t if S $\gamma_\emptyset$-connects $P(T_t) \setminus \{\ell\}$ and additionally the following conditions hold.

- If $\ell = \gamma_X$ then S also contains a path from t to x.
- If $\ell = p$ for some $p \in P(T_t)$ then:
  - If $p \cap V(T_t) = \{a\}$ then S contains a t-a path disjoint from x.

- If $p \cap V(T_t) = \{a, b\}$ then S contains a t-a path disjoint from x and a b-x path disjoint from a or S contains a t-b path disjoint from x and an a-x path disjoint from b.

For $\ell \in \{\gamma_\emptyset, \gamma_X\} \cup P(T_t)$, we say that T_t is ℓ-connected if there is a set S which ℓ-connects $P(T_t)$ in $G[T_t \cup X]$. See also Figure 1 for an illustration of these notions.

Informally, a tree T_t is:

- $\gamma_\emptyset$-connected if all its terminal pairs in $P(T_t)$ can be connected in $G[T_t \cup X]$ either to themselves or to x,
- γ_X-connected if it is $\gamma_\emptyset$-connected and additionally there is a path from its root to x (which can later be used to connect some terminal not in T_t to x via the root of T),
- γ_p-connected if all but one of its terminals, i.e., one of the terminals in p, can be connected in $G[T_t \cup X]$ either to themselves or to x, and additionally one terminal in p can be connected to the root of T_t (from which it can later be connected to x or the other terminal in p).

The following lemma shows that an instance has a solution if and only if every tree T of $G - X$ is $\gamma_\emptyset$-connected. An example for a set of edge-disjoint paths witnessing that a tree T is $\gamma_\emptyset$-connected is illustrated in Figure 2.

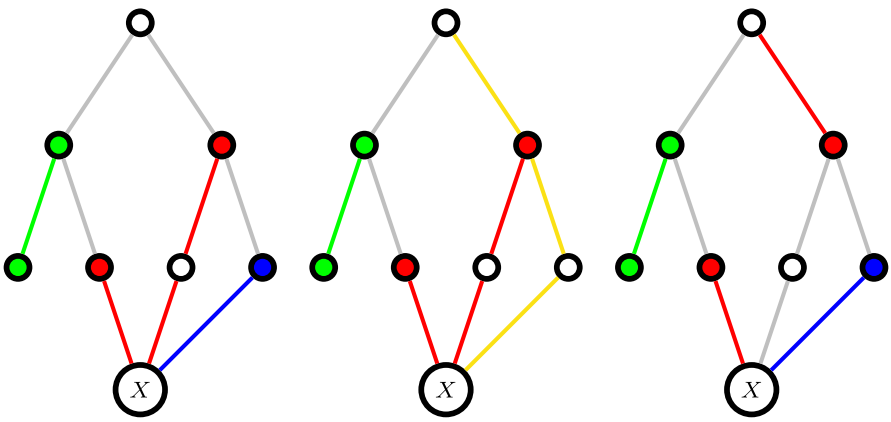

Fig. 1 Illustration of the connectedness notions introduced in Definition 3. Colored vertices (i.e., using the colors green, red, and blue) represent terminal pairs and two vertices having the same color belong to the same terminal pair. Colored edges (i.e., using the colors green, red, blue, and yellow) represent edges used by edge-disjoint paths in S. For instance, the left-most tree has three terminal pairs with the blue terminal pair occuring only once. The colored edges (of the left-most tree) show that the tree is $\gamma_\emptyset$-connected since every terminal is either connected to X or to its counterpart. The colored edges of the tree in the center show that the tree is γ_X-connected. Here the yellow edges give the path from the root of the tree to X. Finally, the colored edges in the right tree show that the tree is p-connected, where p is the terminal pair for the red vertices (Color figure online)

Lemma 1 *(G, X, P) has a solution if and only if every tree T in $G - X$ is $\gamma_\emptyset$-connected.*

Proof Let S be a solution for (G, X, P), i.e., a set of pairwise edge disjoint paths between all terminal pairs in P. Consider the set S' of pairwise edge disjoint paths obtained from S after splitting all paths in S between two terminals a and b that intersect x, into two paths, one from a to x and the other from x to b. Then the restriction of S' to any tree T in $G - X$ shows that T is $\gamma_\emptyset$-connected.

In the converse direction, for every tree T in $G - X$, let S_T be a set of pairwise edge disjoint paths witnessing that T is $\gamma_\emptyset$-connected. Consider a set $p = \{a, b\} \in P$ and let T_a and T_b be the tree containing a and b, respectively. If $T_a = T_b$ then either S_{T_a} contains an a-b path or S_{T_a} contains an a-x path and a b-x path. In both cases we obtain an a-b path in G. Similarly, if $T_a \neq T_b$ then S_{T_a} contains an a-x path and S_{T_b} contains a b-x path, whose concatenation gives us an a-b path in G. Since the union of all S_T over all trees T in $G - X$ is a set of pairwise edge disjoint paths, this shows that (G, X, P) has a solution. This completes the proof of the lemma. □

Due to Lemma 1, our algorithm to solve EDP only has to determine whether every tree in $G - X$ is $\gamma_\emptyset$-connected. For a tree T in $G - X$, our algorithm achieves this by computing a set of labels $L(t)$, where $L(t)$ is the set of all labels $\ell \in \{\gamma_\emptyset, \gamma_X\} \cup P(T_t)$ such that T_t is ℓ-connected, via a bottom-up dynamic programming procedure. We begin by arguing that for a leaf vertex l, the value $L(l)$ can be computed in constant time.

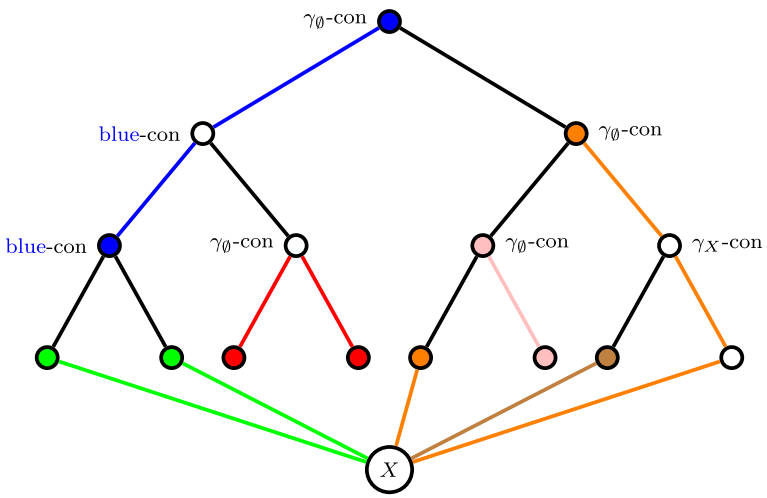

Fig. 2 An Illustration of a solution for a tree. The colored vertices represent terminal pairs with two terminals having the same color if they belong to the same terminal pair. The solution S is given by the colored edges that represent the path connecting the terminal pairs of the respective colors. The label at each tree node says what connectness property is satisfied by the tree rooted at the node (w.r.t. the illustrated solution S) (Color figure online)

Lemma 2 *The set $L(l)$ for a leaf vertex l of T can be computed in time $\mathcal{O}(1)$.*

Proof Since l is a leaf vertex, we conclude that T_l is $\gamma_\emptyset$-connected if and only if either $l \notin \widetilde{P}$ or $l \in \widetilde{P}$ and $(l, x) \in E(G)$. Similarly, T_l is γ_X-connected if and only if $l \notin \widetilde{P}$ and $(l, x) \in E(G)$. Finally, T_l is ℓ-connected for some $\ell \in P(T_l)$ if and only if $l \in \widetilde{P}$. Since all these properties can be checked in constant time, the statement of the lemma follows. □

We will next show how to compute $L(t)$ for a non-leaf vertex $t \in V(T)$ with children $t_1, \ldots, t_l$.

Definition 4 We define the following three sets.

$$V_t^{\neg\gamma_\emptyset} = \{\, t_i \mid \gamma_\emptyset \notin L(t_i) \mid 1 \le i \le l \,\}$$
$$V_t^{\gamma_X} = \{\, t_i \mid \gamma_X \in L(t_i) \mid 1 \le i \le l \,\}$$
$$V_t = \{t_1, \ldots, t_l\} \setminus \left(V_t^{\neg\gamma_\emptyset} \cup V_t^{\gamma_X}\right)$$

That is, $V_t^{\neg\gamma_\emptyset}$ is the set of those children t_i, $1 \le i \le l$, such that T_i is *not* $\gamma_\emptyset$-connected, $V_t^{\gamma_X}$ is the set of those children t_i such that T_i is γ_X-connected and V_t is the set comprising the remaining children. Observe that $\{V_t, V_t^{\neg\gamma_\emptyset}, V_t^{\gamma_X}\}$ forms a partition of $\{t_1, \ldots, t_l\}$ and moreover $\gamma_\emptyset \in L(t)$ and $\gamma_X \notin L(t)$ for every $t \in V_t$. Let $H(t)$ be the graph with vertex set $V_t \cup V_t^{\neg\gamma_\emptyset}$ having an edge between t_i and t_j (for $i \neq j$) if and only if $L(t_i) \cap L(t_j) \neq \emptyset$ and not both t_i and t_j are in V_t. The following lemma is crucial to our algorithm, because it provides us with a simple characterization of $L(t)$ for a non-leaf vertex $t \in V(T)$. See also Figure 3 for an illustration of $H(t)$, the sets V_t, $V_t^{\neg\gamma_\emptyset}$, and $V_t^{\gamma_X}$, as well as the characterization given in the following lemma.

Lemma 3 *Let t be a non-leaf vertex of T with the children $t_1, \ldots, t_l$. Then T_t is:*

- $\gamma_\emptyset$-connected if and only if $L(t') \neq \emptyset$ for every $t' \in \{t_1, \ldots, t_l\}$ and $H(t)$ has a matching M such that $|V_t^{\neg\gamma_\emptyset} \setminus V(M)| \le |V_t^{\gamma_X}|$,
- γ_X-connected if and only if $L(t') \neq \emptyset$ for every $t' \in \{t_1, \ldots, t_l\}$ and $H(t)$ has a matching M such that $|V_t^{\neg\gamma_\emptyset} \setminus V(M)| < |V_t^{\gamma_X}|$,
- ℓ-connected (for $\ell \in P(T_t)$) if and only if $L(t') \neq \emptyset$ for every $t' \in \{t_1, \ldots, t_l\}$ and there is a t_i, $1 \le i \le l$, with $\ell \in L(t_i)$ such that $H(t) - \{t_i\}$ has a matching M with $|V_t^{\neg\gamma_\emptyset} \setminus V(M)| \le |V_t^{\gamma_X}|$.

Proof Towards showing the forward direction let S be a set of pairwise edge disjoint paths witnessing that T_t is ℓ-connected for some $\ell \in \{\gamma_\emptyset, \gamma_X\} \cup P(T_t)$. Then S contains the following types of paths:

(T1) A path between a and b that does not contain x, where $\{a, b\} \in P(T_t)$,
(T2) A path between a and x, which does not contain b, where $\{a, b\} \in P(T_t)$,
(T3) A path between x and t (only if $\ell = \gamma_X$),

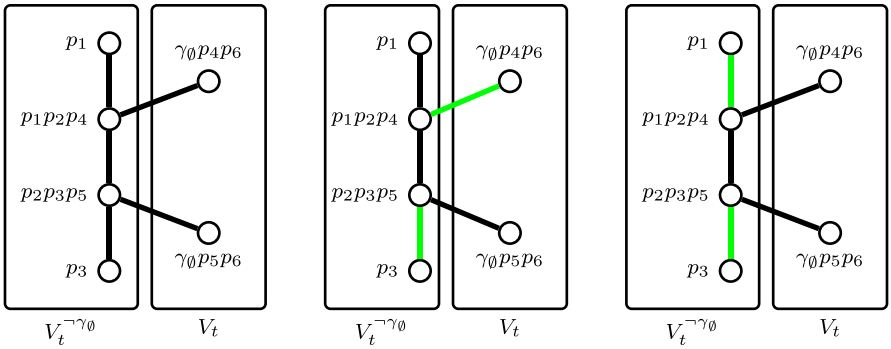

Fig. 3 An Illustration of the graph $H(t)$ together with the matchings M required for the characterization given Lemma 3. The example assumes that t has 7 children with 4 or them being in $V_t^{\neg\gamma_\emptyset}$, two of them being in V_t, and one child in $V_t^{\gamma_X}$. The figure only shows the children in $V_t^{\neg\gamma_\emptyset} \cup V_t$ since these make up the vertex set of the graph $H(t)$. The vertex labels give the set $L(t_i)$ for each of the children of t and the edges give the edges in $H(t)$. The left-most figure illustrates $H(t)$. The middle figure shows a matching M (the green edges) that shows that T_t is $\gamma_\emptyset$-connected. The right figure shows a matching M that shows that T_t is γ_X-connected

(T4) A path between a and t, which does not contain x, (only if $\ell = p$ for some $p \in P(T_t)$ and $a \in p$),

For every i with $1 \leq i \leq l$, let S_i be the subset of S containing all paths that use at least one vertex in T_{t_i} and let S_i' be the restriction of all paths in S_i to $G[T_{t_i} \cup X]$. Consider an i with $1 \leq i \leq l$. Because the paths in S are pairwise edge disjoint, we obtain that at most one path in S contains the edge (t_i, t). We start with the following observations.

(O1) If S_i contains no path that contains the edge (t_i, t), then S_i' shows that T_{t_i} is $\gamma_\emptyset$ -connected.
(O2) If S_i contains a path say s_i that contains the edge (t_i, t), then the following statements hold.

  (O21) If s_i is a path of Type (T1) for some $p \in P(T_t)$, then S_i' shows that T_{t_i} is p-connected,
  (O22) If s_i is a path of Type (T2) for some $p \in P(T_t)$, then the following statements hold.

    (O221) If the endpoint of s_i $G[T_{t_i} \cup X$ is a terminal in $a \in p$ then s_i $G[T_{t_i}$ is p-connected.
    (O222) Otherwise, i.e., if the endpoint of s_i $G[T_{t_i}$ is x, then s_i shows that T_{t_i} is γ_X-connected.

(O3) If s_i is a path of Type (T3), then S_i' shows that T_{t_i} is γ_X-connected.
(O4) If s_i is a path of Type (T4), for some terminal-pair $p \in P(T_{t_i})$ then S_i' shows that T_{t_i} is p-connected.

Let M be the set of all pairs $\{t_i, t_j\} \subseteq V_t \cup V_t^{\neg\gamma_\emptyset}$ such that $t_i \in V_t^{\neg\gamma_\emptyset}$ or $t_j \in V_t^{\neg\gamma_\emptyset}$ and S contains a path that contains both edges (t_i, t) and (t, t_j). We claim that M is a matching of $H(t)$. Since an edge (t_i, t) can be used by at most one path in S, it follows that the pairs in M are pairwise disjoint and it remains to show that $M \subseteq E(H(t))$, namely, $L(t_i) \cap L(t_j) \neq \emptyset$ for every $(t_i, t_j) \in M$. Let $s \in S$ be the path witnessing that $(t_i, t_j) \in M$. Because s contains the edges (t_i, t) and (t, t_j), it cannot be of Type (T3) or (T4). Moreover, if s is of Type (T2) then either T_{t_i} or T_{t_j} is γ_X-connected, contradicting our assumption that $t_i, t_j \in V_t \cup V_t^{\neg\gamma_\emptyset}$. Hence s is of Type (T1) for some terminal-pair $p \in P(T_t)$, which implies that T_{t_i} and T_{t_j} are p-connected, as required.

In the following let $t_i \in V_t^{\neg\gamma_\emptyset}$. Because of Observation (O1), we obtain that S_i contains a path, say s_i, using the edge (t_i, t) (otherwise T_{t_i} is $\gamma_\emptyset$-connected). Moreover, together with Observations (O2)–(O4), we obtain that either:

(P1) s_i is a path of Type (T1) for some $p \in P(T_t)$,
(P2) s_i is a path of Type (T2) for some $p \in P(T_t)$ and the endpoint of s_i in $G[T_{t_i} \cup X]$ is a terminal $a \in p$,
(P3) s_i is a path of Type (T4) for some $p \in P(T_t)$.

Because t is not connected to x and t is not a terminal, we obtain that if s_i satisfies (P1), then there is a $t_j \in \{t_1, \ldots, t_l\}$ with $p \in L(t_j)$ such that s_i contains the edge (t_j, t). Similarly, if s_i satisfies (P2) then there is a $t_j \in V_t^{\gamma_X}$ such that s_i contains the edge (t_j, t). Consequently, every $t_i \in V_t^{\neg\gamma_\emptyset}$ for which s_i satisfies (P1) or (P2) is mapped to a unique $t_j \in \{t_1, \ldots, t_l\}$ such that s_i contains the edge (t_j, t).

We now distinguish three cases depending on ℓ. If $\ell = \emptyset$, then S contains no path of type (T4) and hence for every $t_i \in V_t^{\neg\gamma_\emptyset}$ either (P1) or (P2) has to hold. In particular, for every $t_i \in V_t^{\neg\gamma_\emptyset} \setminus V(M)$ there must be a $t_j \in V_t^{\gamma_X}$ such that s_i contains the edge (t_j, t). Consequently, if $|V_t^{\gamma_X}| < |V_t^{\neg\gamma_\emptyset} \setminus V(M)|$, this is not achievable contradicting our assumption that S is a witness to the fact that T_t is $\gamma_\emptyset$-connected.

If $\ell = \gamma_X$, then S contains a path of Type (T3), which due to Observation (O3) uses the edge (t', t) for some $t' \in V_t^{\gamma_X}$. Since again (P1) or (P2) has to hold for every $t_i \in V_t^{\neg\gamma_\emptyset}$, we obtain that for every $t_i \in V_t^{\neg\gamma_\emptyset} \setminus V(M)$ there must be a $t_j \in V_t^{\gamma_X} \setminus \{t'\}$ such that s_i contains the edge (t_j, t). Consequently, if $|V_t^{\gamma_X}| \leq |V_t^{\neg\gamma_\emptyset} \setminus V(M)|$ then $|V_t^{\gamma_X} \setminus \{t_j\}| < |V_t^{\neg\gamma_\emptyset} \setminus V(M)|$ and this is not achievable contradicting our assumption that S is a witness to the fact that T_t is γ_X-connected.

Finally, if $\ell = p$ for some $p \in P(T_t)$, then S contains a path of Type (T4), which due to Observation (O4) uses the edge (t', t) for some $t' \in \{t_1, \ldots, t_l\}$ with $p \in L(t')$. Observe that t' cannot be matched by M and hence M is also a matching of $H(t) - \{t'\}$. Moreover, Since (P1) or (P2) has to hold for every $t_i \in V_t^{\neg\gamma_\emptyset} \setminus \{t'\}$, we obtain that for every $t_i \in V_t^{\neg\gamma_\emptyset} \setminus (V(M) \setminus \{t'\})$ there must be a $t_j \in V_t^{\gamma_X} \setminus \{t'\}$ such that s_i contains the edge (t_j, t). Consequently, if $|V_t^{\gamma_X} \setminus \{t'\}| < |V_t^{\neg\gamma_\emptyset} \setminus (V(M) \cup \{t'\})|$ then this is not achievable contradicting our assumption that S is a witness to the fact that T_t is p-connected.

Towards showing the reverse direction we will show how to construct a set S of pairwise edge disjoint paths witnessing that T_t is ℓ-connected.

If $\ell = \emptyset$, then let M be a matching of $H(t)$ such that $|V_t^{\neg\gamma_\emptyset} \setminus V(M)| \leq |V_t^{\gamma_X}|$ and let α be a bijection between $V_t^{\neg\gamma_\emptyset} \setminus V(M)$ and $|V_t^{\neg\gamma_\emptyset} \setminus V(M)|$ arbitrarily chosen elements in $V_t^{\gamma_X}$. Then the set S of pairwise edge disjoint paths witnessing that T_t is $\gamma_\emptyset$-connected is obtained as follows.

(S1) For every $t_i \in V_t \setminus V(M)$, let S' be a set of pairwise edge disjoint paths witnessing that T_{t_i} is $\gamma_\emptyset$-connected. Add all paths in S' to S.
(S2) For every $t_i \in V_t^{\gamma_X} \setminus \{ \alpha(t') \mid t' \in V_t^{\neg\gamma_\emptyset} \setminus V(M) \}$, let S' be a set of pairwise edge disjoint paths witnessing that T_{t_i} is $\gamma_\emptyset$-connected. Add all paths in S' to S.
(S3) For every $(t_i, t_j) \in M$, choose $p \in L(t_i) \cap L(T_j) \cap P(T_t)$ arbitrarily and let S_1 and S_2 be sets of pairwise edge disjoint paths witnessing that T_{t_i} and T_{t_j} are p-connected, respectively. Moreover, let $s_1 \in S_1$ and $s_2 \in S_2$ be the unique paths connecting a terminal in p with t_i and t_j, respectively. Then add all path in $(S_1 \setminus \{s_1\}) \cup (S_2 \setminus \{s_2\})$ and additionally the path $s_1 \circ (t_i, t, t_j) \circ s_2$ to S.
(S4) For every $t_i \in V_t^{\neg\gamma_\emptyset} \setminus V(M)$, choose $p \in L(t_i) \cap P(T_t)$ arbitrarily and let S_1 be a set of pairwise edge disjoint paths witnessing that T_{t_i} is p-connected. Moreover, let s_1 be the unique path in S_1 connecting a terminal in p with t_i. Furthermore, let S_2 be a set of pairwise edge disjoint paths witnessing that $T_{\alpha(t_i)}$ is γ_X-connected and let s_2 be the unique path in S_2 connecting x with $\alpha(t_i)$. Then add all path in $(S_1 \setminus \{s_1\}) \cup (S_2 \setminus \{s_2\})$ and additionally the path $s_1 \circ (t_i, t, \alpha(t_i)) \circ s_2$ to S.

It is straightforward to verify that S witnesses that T_t is $\gamma_\emptyset$-connected.

If $\ell = \gamma_X$, then the set S of pairwise edge disjoint paths witnessing that T_t is γ_X -connected is defined analogously to the case where $\ell = \emptyset$ only that now step (S2) is replaced by:

(S2') Choose one $t' \in V_t^{\gamma_X} \setminus \{ \alpha(t') \mid t' \in V_t^{\neg\gamma_\emptyset} \setminus V(M) \}$ arbitrarily; note that such a t' must exist because $|V_t^{\neg\gamma_\emptyset} \setminus V(M)| < |V_t^{\gamma_X}|$. Let S' be a set of pairwise edge disjoint paths witnessing that $T_{t'}$ is γ_X-connected and let s' be the unique path in S' connecting t' with x. Then add all paths in $S' \setminus s'$ to S and additionally the path obtained from s' after adding the edge (t', t). Finally, for every child in $V_t^{\gamma_X} \setminus (\{ \alpha(t'') \mid t'' \in V_t^{\neg\gamma_\emptyset} \setminus V(M) \} \cup \{t'\})$ proceed as described in Step (S2).

Finally, if $\ell = p$ for some $p \in P(T_t)$, then let t_i be a child with $p \in L(t_i)$ and let M be a matching of $H(t) - \{t_i\}$ such that $|V_t^{\neg\gamma_\emptyset} \setminus (V(M) \cup \{t_i\})| \leq |V_t^{\gamma_X} \setminus \{t_i\}|$. Moreover, and let α be a bijection between $V_t^{\neg\gamma_\emptyset} \setminus (V(M) \cup \{t_i\})$ and $|V_t^{\neg\gamma_\emptyset} \setminus (V(M) \cup \{t_i\})|$ arbitrarily chosen elements in $V_t^{\gamma_X} \setminus \{t_i\}$. Then the set S of pairwise edge disjoint paths witnessing that T_t is p-connected is obtained analogously to the case where $\ell = \emptyset$ above only that we do not execute the steps (S1)–(S4) for t_i but instead do the following:

(S0) Let S' be a set of pairwise edge disjoint paths witnessing that T_{t_i} is p-connected and let s' be the unique path in S' connecting a terminal in p to t_i. Then we add all path in $S' \setminus \{s'\}$ and additionally the path $s' \circ (t_i, t)$ to S.

This completes the proof of Lemma 3. □

The following two lemmas show how the above characterization can be employed to compute $L(t)$ for a non-leaf vertex t of T. Since the matching employed in Lemma 3 needs to maximize the number of vertices covered in $V_t^{\neg\gamma_\emptyset}$, we first show how such a matching can be computed efficiently.

Lemma 4 *There is an algorithm that, given a graph G and a subset S of $V(G)$, computes a matching M maximizing"$> |V(M) \cap S|$ in time $\mathcal{O}(\sqrt{|V|}|E|)$.*

Proof We reduce the problem to Maximum Weighted Matching problem, which can be solved in time $\mathcal{O}(\sqrt{|V|}|E|)$ [25]. The reduction simply assigns weight two to every edge that is completely contained in S, weight one to every edge between S and $V(G)\backslash S$, and weight zero otherwise. The correctness of the reduction follows because the weight of any matching M in the weighted graph equals the number of vertices in S covered by M. □

Lemma 5 *Let t be a non-leaf vertex of T with children $t_1, \ldots, t_l$. Then $L(t)$ can be computed from $L(t_1), \ldots, L(t_l)$ in time $\mathcal{O}(|P(T_t)|l^2\sqrt{l})$.*

Proof First we construct the graph $H(t)$ in time $\mathcal{O}(l^2|P(T_t)|)$. We then check for every $\ell \in \{\gamma_\emptyset, \gamma_X\} \cup P(T_t)$, whether T_t is ℓ-connected with the help of Lemma 3 as follows. If $\ell \in \{\gamma_\emptyset, \gamma_X\}$ we compute a matching M in $H(t)$ that maximizes $|V(M) \cap V_t^{\neg\gamma_\emptyset}|$, i.e., the number of matched vertices in $V_t^{\neg\gamma_\emptyset}$. This can be achieved according to Lemma 4 in time $\mathcal{O}(\sqrt{l}l^2)$. Then in accordance with Lemma 3, we add $\gamma_\emptyset$ or γ_X to $L(t)$ if $|V_t^{\neg\gamma_\emptyset}\backslash V(M)| \leq |V_t^{\gamma_X}|$ or $|V_t^{\neg\gamma_\emptyset}\backslash V(M)| \leq |V_t^{\gamma_X}|$, respectively. For any $\ell \in P(T_t)$, again in accordance with Lemma 3, we compute for every child t' of t with $\ell \in P(T_{t'})$, a matching M in $H(t) - \{t'\}$ that maximizes $|V(M) \cap V_t^{\neg\gamma_\emptyset}|$. Since t has at most two children t' with $\ell \in P(T_{t'})$ and due to Lemma 4 such a matching can be computed in time $\mathcal{O}(\sqrt{l}l^2)$, this is also the total running time for this step of the algorithm. If one of the at most two such matchings M satisfies $|V_t^{\neg\gamma_\emptyset}\backslash(V(M) \cup \{t'\})| \leq |V_t^{\gamma_X}\backslash\{t'\}|$, we add ℓ to $L(t)$ and otherwise we do not. This completes the description of the algorithm whose total running time can be obtained as the time required to construct $H(t)$ ($\mathcal{O}(l^2|P(T_t)|)$ plus $|P(T_t)| + 2$ times the time required to compute the required matching in $H(t)$ ($\mathcal{O}(\sqrt{l}l^2)$), which results in a total running time of $\mathcal{O}(l^2|P(T_t)| + |P(T_t)|\sqrt{l}l^2) = O(|P(T_t)|l^2\sqrt{l})$. □

We are now ready to put everything together into an algorithm that decides whether a tree T is $\gamma_\emptyset$-connected.

Lemma 6 *Let T be a tree in $G - X$. There is an algorithm that decides whether T is $\gamma_\emptyset$-connected in time $\mathcal{O}(|P(T)||V(T)|^{\frac{5}{2}})$.*

Proof The algorithm computes the set of labels $L(t)$ for every vertex $t \in V(T)$ using a bottom-up dynamic programming approach. Starting from the leaves of T, for which the set of labels can be computed due to Lemma 2 in constant time, it uses

Lemma 5 to compute $L(t)$ for every inner node t of T in time $\mathcal{O}(|P(T_t)|l^2\sqrt{l})$. The total running time of the algorithm is then the sum of the running time for any inner node of T plus the number of leaves of T, i.e., $\mathcal{O}(|P(T)||V(T)|^{\frac{5}{2}})$. □

Theorem 1 SEDP *can be solved in time* $\mathcal{O}(|P||V(G)|^{\frac{5}{2}})$.

Proof We first employ Lemma 6 to determine whether every tree T of $G - X$ is $\gamma_\emptyset$ -connected. If so we output Yes and otherwise No. Correctness follows from Lemma 1. □

4 Treewidth and Maximum Degree

The goal of this section is to obtain an FPT-algorithm for EDP parameterized by the treewidth ω and maximum degree Δ of the input graph. Before we proceed to the theorem, we remark that the transformation described in Sect. 2.2 (which ensures that each terminal occurs in a leaf of G and that no vertex occurs in more than one terminal pair) does not increase the treewidth of the augmented graph by more than 2—indeed, the transformation merely results in the replacement of each edge in G^P (connecting two terminals in P) with a path containing two new internal vertices (the leaves created in G for these two terminals). It is well known (and easy to see) that subdividing a set of edges in a graph twice can only increase treewidth by at most 2.

Theorem 2 EDP can be solved in time $2^{\mathcal{O}(\Delta\omega\log\omega)} \cdot n$, where ω, Δ and n are the treewidth, maximum degree and number of vertices of the input graph G, respectively.

Proof Let (G, P) be an instance of EDP and let $(T, \mathcal{B})$ be a nice tree-decomposition of G of width at most $k = 5\omega + 4$; recall that such $(T, \mathcal{B})$ can be computed in time $2^{\mathcal{O}(k)} \cdot n$ by Fact 1. Consider the following leaf-to-root dynamic programming algorithm Å, executed on T. At each bag B_t associated with a node t of T, Å will compute a table $\mathcal{M}_t$ of *records*, which are tuples of the form $\{(\mathsf{used}, \mathsf{give}, \mathsf{single})\}$ where:

- used is a multiset of subsets of B_t of cardinality 2 such that $\forall v \in B_t : |\{ U \in B_T \mid v \in U \}| \leq \Delta$ (i.e., v occurs in at most Δ-many subsets in used).
- give is a mapping from subsets of B_t of cardinality 2 to $[\Delta]$ such that $\forall v \in B_t : (\sum_{u \in B_t} \mathsf{give}(\{v, u\})) \leq \Delta$ (i.e., the sets containing v receive a total value of at most Δ), and
- single is a mapping which maps each terminal $a_i \in Y_t$ such that its counterpart $b_i \notin Y_t$ to an element of B_t.

Before we proceed to describe the steps of the algorithm itself, let us first introduce the semantics of a record. For a fixed t, we will consider the graph G_t

obtained from $G[Y_t]$ by removing all edges with both endpoints in B_t (we note that this "pruned" definition of G_t is not strictly necessary for the algorithm, but makes certain steps easier later on). Then $\alpha = \{(\mathsf{used}, \mathsf{give}, \mathsf{single})\} \in \mathcal{M}_t$ if and only if there exists a set of edge disjoint paths Q in G_t and a surjective mapping τ from terminal pairs occurring in Y_t to subsets of B_t of size two with the following properties:

- For each terminal pair ab that occurs in Y_t:
  - Q either contains a path whose endpoints are a and b, or
  - Q contains an a-x_1 path for some $x_1 \in B_t$ and a b-x_2 path for some $x_2 \in B_t$ which is distinct from x_1, and furthermore $\tau(ab) = \{x_1, x_2\} \in \mathsf{used}$;
- for each terminal pair ab such that $a \in Y_t$ but $b \notin Y_t$:
  - Q contains a path whose endpoints are a and $x \in B_t$, where $(a, x) \in \mathsf{single}$;
- for each distinct $x_1, x_2 \in B_t$, Q contains precisely $\mathsf{give}(\{x_1, x_2\})$ paths from x_1 to x_2.

In the above case, we say that Q witnesses α. It is important to note that the equivalence between the existence of records and sets Q of pairwise edge disjoint paths only holds because of the bound on the maximum degree. That is because every vertex of G has degree at most Δ, it follows that any set Q of pairwise edge disjoint paths can contain at most Δ paths containing a vertex in the boundary. Moreover, we note that by reversing the above considerations, given a set of edge disjoint paths Q in G_t satisfying a certain set of conditions, we can construct in time $3^{\Delta k}$ a set of records in $\mathcal{M}_t$ that are witnessed by Q (one merely needs to branch over all options of assigning paths in α which end in the boundary: they may either contribute to give or to single or to used). These conditions are that each path either (i) connects a terminal pair, (ii) connects a terminal pair to two vertices in B_t, (iii) connects two vertices in B_t, or (iv) connects a terminal $a \in Y_t$ whose counterpart b does not lie in Y_t to a vertex in B_t.

Å runs as follows: it begins by computing the records $\mathcal{M}_t$ for each leaf t of T. It then proceeds to compute the records for all remaining nodes in T in a bottom-up fashion, until it computes $\mathcal{M}_r$. Since $B_r = \emptyset$, it follows that (G, P) is a yes-instance if and only if $(\emptyset, \emptyset, \emptyset) \in \mathcal{M}_r$. For each record α, it will keep (for convenience) a set Q_α of edge disjoint paths witnessing α. Observe that while for each specific α there may exist many possible choices of Q_α, all of these interact with B_t in the same way.

We make one last digression before giving the procedures used to compute $\mathcal{M}_t$ for the four types of nodes in nice tree-decompositions. First, observe that number of edge disjoint paths in G_t ending in B_t is upper-bounded by Δk, it follows that each record in $\mathcal{M}_t$ satisfies $|\mathsf{single}| \leq \Delta k$. The total number of possible records is upper-bounded by all possible choices for single ($2^{\Delta \cdot k}$), times the number of choices for used (which is $2^{\Delta \cdot k \log k}$: there are at most k^Δ choices for each vertex, and hence at most $(k^\Delta)^k$ choices in total), times give (which can be argued equivalently as used, since it is merely a different encoding of a multiset). As a consequence, $|\mathcal{M}_t| \leq 2^{\mathcal{O}(\Delta k \log k)}$

for each node t in T, which is crucial to obtain an upper-bound on the running time of Å.

Case 1: *t is a leaf node.* If $v \in B_t$ is not a terminal, then $\mathcal{M}_t = \{\emptyset, \emptyset, \emptyset\}$. On the other hand, if $v \in B_t$ is a terminal, then $\mathcal{M}_t = \{\emptyset, \emptyset, \{(v, v)\}\}$.

Case 2: *t is an introduce node:* Let p be the child of t in T and let v be the vertex introduced at t, i.e., $v \in B_t \backslash B_p$. Recall that v has no neighbors in G_t. As a consequence, if v is not a terminal, then $\mathcal{M}_t = \mathcal{M}_p$. If v is a terminal and its counterpart w lies in G_t, then for each $\{(\mathsf{used}, \mathsf{give}, \mathsf{single})\} \in \mathcal{M}_p$ we add the record $\{(\mathsf{used}', \mathsf{give}', \mathsf{single}')\}$ to $\mathcal{M}_t$, where:

- $\mathsf{give}' = \mathsf{give}$,
- single' is obtained by deleting the unique tuple $(w, ?)$ from single, and
- used' is obtained by adding the subset $\{v, ?\}$ to used.

Finally, if v is a terminal and its counterpart w does not lie in G_t, then for each $\{(\mathsf{used}, \mathsf{give}, \mathsf{single})\} \in \mathcal{M}_p$ we add the record $\{(\mathsf{used}, \mathsf{give}, \mathsf{single} \cup (v, v))\}$ to $\mathcal{M}_t$.

Case 3: *t is a join node:* Let p, q be the two children of t in T. For each $\{(\mathsf{used}_p, \mathsf{give}_p, \mathsf{single}_p)\} \in \mathcal{M}_p$ and each $\{(\mathsf{used}_q, \mathsf{give}_q, \mathsf{single}_q)\} \in \mathcal{M}_p$, we add a new record $\{(\mathsf{used}, \mathsf{give}, \mathsf{single})\}$ to $\mathcal{M}_t$ constructed as follows:

1. for each $x, y \in B_t$, we set $\mathsf{give}(xy) := \mathsf{give}_p(xy) + \mathsf{give}_q(xy)$;
2. for each terminal pair v, w such that $(v, ?_v) \in \mathsf{single}_p$ and $(w, ?_w) \in \mathsf{single}_q$ where $?_v = ?_w$,
   - we delete $(v, ?_v)$ from single_p and
   - we delete $(w, ?_w)$ from single_q;
3. for each terminal pair v, w such that $(v, ?_v) \in \mathsf{single}_p$ and $(w, ?_w) \in \mathsf{single}_q$ where $?_v \neq ?_w$,
   - we delete $(v, ?_v)$ from single_p and
   - we delete $(w, ?_w)$ from single_q and
   - we add $\{?_v, ?_w\}$ to used;
4. we set $\mathsf{used} := \mathsf{used} \cup \mathsf{used}_p \cup \mathsf{used}_q$;
5. we set $\mathsf{single} := \mathsf{single}_p \cup \mathsf{single}_q$;
6. finally, we restore the records in $\mathcal{M}_p$ and $\mathcal{M}_q$ to their original state as of step 1 (i.e., we restore all deleted items).

Case 4: *t is a forget node:* Let p by the child of t in T and let v be the vertex forgotten at t, i.e., $v \in B_p \backslash B_t$. We note that this will be the by far most

complicated step, since we are forced to account for the edges between v and its neighbors in B_t.

Let us begin by considering how the records in $\mathcal{M}_t$ can be obtained from those in $\mathcal{M}_p$; in particular, let us fix an arbitrary $\alpha = (\mathsf{used}_t, \mathsf{give}_t, \mathsf{single}_t) \in \mathcal{M}_t$. This means that there exists a set Q_α of edge disjoint paths (along with a mapping τ) in G_t satisfying the conditions given by α. Furthermore, let $E_v = \{\, wv \in E(G) \mid w \in B_t \,\}$, i.e., E_v is the set of edges which are not present in G_p but are present in G_t. The crucial observation is that for each set Q_α, there exists a set Q_β of edge disjoint paths satisfying the conditions given by some $\beta = (\mathsf{used}_p, \mathsf{give}_p, \mathsf{single}_p) \in \mathcal{M}_p$ such that Q_α can be obtained by merely "extending" Q_β using the edges in E_v; in particular, E_v can increase the number of paths contributing to give_p, change the endpoints of paths captured by used_p, and also change the endpoints of paths captured by single_p.

Formally, we will proceed as follows. For each $\beta \in \mathcal{M}_p$, we begin with the witness Q_β stored by Å, and then branch over all options of how the addition of the edges in E_v may be used to augment the paths in Q_β. In particular, since $|E_v| \leq k$, there are at most $k+1$ vertices incident to an edge in E_v, and due to the degree bound of Δ it then follows that there are at most $\Delta(k+1)$ distinct paths in Q_β which may be augmented by E_v. Since each edge in E_v may either be assigned to extend one such path or form a new path, we conclude that there exist at most $k^{\mathcal{O}(\Delta k)}$ possible resulting sets of edge-disjoint paths in G_t. For each such set Q' of edge-disjoint paths, we then check that conditions (i)-(iv) for constructing records witnessed by Q' hold, and in the positive case we construct these records in time $3^{\Delta k}$ as argued earlier and add them to $\mathcal{M}_t$.

Summary and running time. Algorithm Å begins by invoking Fact 1 to compute a tree-decomposition of width at most $k = 5\omega + 4$ and $\mathcal{O}(n)$ nodes, and then converts this into a nice tree-decomposition $(T, \mathcal{B})$ of the same width and also $\mathcal{O}(n)$ nodes. It then proceeds to compute the records $\mathcal{M}_t$ (along with corresponding witnesses) for each node t of T in a leaves-to-root fashion, using the procedures described above. The number of times any procedure is called is upper-bounded by $\mathcal{O}(n)$, and the running time of every procedure is upper-bounded by the worst-case running time of the procedure for forget nodes. There, for each record β in the data table of the child of t, the algorithm takes its witness Q_β and uses branching to construct at most $k^{\mathcal{O}(\Delta k)}$ new witnesses (after the necessary conditions are checked). Each such witness Q_α gives rise to a set of records that can be computed in time $3^{\Delta k}$, which are then added to $\mathcal{M}_t$ (unless they are already there). All in all, the running time of this procedure is upper-bounded by $2^{\mathcal{O}(\Delta k \log k)} \cdot k^{\mathcal{O}(\Delta k)} \cdot 3^{\Delta k} = 2^{\mathcal{O}(\Delta k \log k)}$, and the run-time of the algorithm follows. □

5 Lower Bounds of EDP for Parameters of the Augmented Graph

In this section we will show that EDP parameterized by the feedback vertex set number (and hence also parameterized by the treewidth) of the augmented graph is W[1]-hard. This nicely complements the result in [29] showing that EDP is solvable in polynomial time for bounded treewidth and also provides a natural justification for the FPT-algorithm presented in the next section, since it shows that more general parameterizations such as treewidth are unlikely to lead to an FPT-algorithm for EDP.

The reduction will proceed in several steps, the first of which uses a reduction from the following multidimensional variant of subset sum, which was introduced in [15].

Multidimensional Relaxed Subset Sum (MRSS)	
Input:	An integer k, a set $S = \{s_1, \ldots, s_n\}$ of item-vectors with $s_i \in \mathbb{N}^k$ for every i with $1 \leq i \leq n$, a target vector $t \in \mathbb{N}^k$, and an integer k'.
Parameter:	$k + k'$
Question:	Is there a subset $S' \subseteq S$ with $\lvert S' \rvert \leq k'$ such that $\sum_{s \in S'} s \geq t$?

We will use the fact that the problem is strongly W[1]-hard, which was shown in [15].

Lemma 7 [15, Lemma 4] MRSS *is* W*[1]-hard even if all integers in the input are given in unary.*

Our next step is to move from subset sum problems to EDP variants, with a final goal of showing the hardness of EDP parameterized by the feedback vertex set number of the augmented graph. We will first show that the following directed version, which also allows for multiple instead of just one path between every terminal pair, is W[1]-hard parameterized by the feedback vertex set number of the augmented graph and the number of terminal pairs.

Multiple Directed Edge Disjoint Paths (MDEDP)	
Input:	A directed graph G, a set P of ℓ triples (s_i, t_i, n_i) with $1 \leq i \leq \ell$, $s_i, t_i \in V(G)$, and $n_i \in \mathbb{N}$.
Question:	Is there a set of pairwise edge disjoint paths containing n_i paths from s_i to t_i for every i with $1 \leq i \leq \ell$?

Lemma 8 *MDEDP is* W*[1]-hard parameterized by the following parameter: the feedback vertex set number of the undirected augmented graph plus the number of*

terminal triples. Furthermore, this holds even for acyclic instances when all integers on the input are given in unary.

Proof We prove the lemma by a parameterized reduction from MRSS. Namely, given an instance $\mathcal{I} = (k, S, t, k')$ of MRSS we construct an equivalent instance $\mathcal{I}' = (G, P)$ of MDEDP in polynomial time such that $|P| \leq k + 1$, $\mathbf{fvs}(\overline{G}^P) \leq 2k + 2$ and G is acyclic.

We start by introducing a gadget $G(z)$ for every item-vector $z \in S$. $G(z)$ is a directed path $(p_1^z, \ldots, p_l^z)$, where $l = 2(\sum_{1 \leq i \leq k} z[i]) + 2$. We let P contain one triple $(s, t, |S| - k')$ as well as one triple $(s_i, t_i, t[i])$ for every i with $1 \leq i \leq k$. Then G is obtained from the disjoint union of $G(z)$ for every $z \in S$ plus the vertices $\{s, t, s_1, t_1, \ldots, s_k, t_k\}$. Moreover, G contains the following edges:

- one edge from s to the first vertex of $G(z)$ for every $z \in S$,
- one edge from the last vertex of $G(z)$ to t for every $z \in S$,
- for every i with $1 \leq i \leq k$ and every $z \in S$ an edge from s_i to the vertex p_j^z (in $G(z)$), where $j = 1 + 2(\sum_{1 \leq j < i} z[j]) + 2l - 1$ for every l with $1 \leq l \leq z[i]$,
- for every i with $1 \leq i \leq k$ and every $z \in S$ an edge from p_j^z (in $G(z)$), where $j = 1 + 2(\sum_{1 \leq j < i} z[j]) + 2l$, to t_i for every l with $1 \leq l \leq z[i]$,

This completes the construction of (G, P). Since $\overline{G}^P - \{s, t, s_1, t_1, \ldots, s_k, t_k\}$ is a disjoint union of directed paths, i.e., one path $G(z)$ for every $z \in S$, we obtain that $\overline{G}^P$ has a feedback vertex set, i.e., the set $\{s, t, s_1, t_1, \ldots, s_k, t_k\}$, of size at most $2(k + 1) = 2k + 2$. Moreover, G is clearly acyclic and $|P| \leq k + 1$. It hence only remains to show that (k, S, t, k') has a solution if and only if so does (G, P).

Towards showing the forward direction let $S' \subseteq S$ be a solution for $\mathcal{I}$. Then we construct a set Q of pairwise edge-disjoint paths in G containing $|S| - k'$ path from s to t as well as $t[i]$ paths from s_i to t_i, i.e., a solution for $\mathcal{I}'$, as follows. For every $z \in S \backslash S'$, Q contains the path $(s, G(z), t)$, which already accounts for the $|S| - k'$ paths from s to t. Moreover, for every i with $1 \leq i \leq k$ and $z \in S'$, Q contains the path $(s_i, p_j^z, p_{j+1}^z, t_i)$ for every j with $1 + 2(\sum_{1 \leq o < i} z[o]) < j \leq 1 + 2(\sum_{1 \leq o < i} z[o]) + 2(z[i])$ and j is even. This concludes the definition of Q. Note that Q now contains $\sum_{z \in S'} z[i]$ paths from s_i to t_i for every i with $1 \leq i \leq k$ and since S' is a solution for $\mathcal{I}$, it holds that $\sum_{z \in S'} z[i] \geq t[i]$. Consequently, Q is a solution for $\mathcal{I}'$.

Towards showing the reverse direction, let Q be a solution for $\mathcal{I}'$. Then Q must contain $|S| - k'$ pairwise edge disjoint paths from s to t. Because every path from s to t in G must have the form $(s, G(z), t)$ for some $z \in S$, we obtain that all the edges of exactly $|S| - k'$ gadgets $G(z)$ for $z \in S$ are used by the paths from s to t in G. Let $S' \subseteq S$ be the set containing all $z \in S$ such that $G(z)$ is not used by a path from s to t in Q. We claim that S' is a solution for $\mathcal{I}$. Clearly, $|S'| = k'$ and it remains to show that $\sum_{z \in S'} z \geq t$. Towards showing this observe that for every i with $1 \leq i \leq k$, a path from s_i to t_i has to use at least one edge $\{p_j^z, p_{j+1}^z\}$ from some $z \in S'$ and j with $1 + 2(\sum_{1 \leq o < i} z[o]) < j \leq 1 + 2(\sum_{1 \leq o \leq i} z[o])$ and j is even. Since $G(z)$ for every $z \in S'$ has at most $z[i]$ of those edges, we obtain that for every $z \in S'$, Q contains at most $z[i]$ path from s_i to t_i using edges in $G(z)$. Because the paths in Q from s_i to t_i cannot

use any edge from $G(z)$ such that $z \notin S'$; this is because all edges of such $G(z)$ are already used by the paths from s to t in Q. Hence the total number of paths in Q from s_i to t_i is at most $\sum_{z \in S'} z[i]$ and since Q contains $t[i]$ paths from s_i to t_i, we obtained that $\sum_{z \in S'} z[i] \geq t[i]$. □

Our next aim is now to reduce from MDEDP to the following undirected version of MDEDP.

Multiple Undirected Edge Disjoint Paths (MUEDP)	
Input:	An undirected graph G, a set P of ℓ triples (s_i, t_i, n_i) with $1 \leq i \leq \ell$, $s_i, t_i \in V(G)$, and $n_i \in \mathbb{N}$.
Question:	Is there a set of pairwise edge disjoint paths containing n_i paths from s_i to t_i for every i with $1 \leq i \leq \ell$?

To do so we first need the following auxiliary lemma.

Lemma 9 *Let $\mathcal{I} = (G, P)$ be an instance of* MDEDP *such that G is acyclic. Then in polynomial time we can construct an instance $\mathcal{I}' = (G', P')$ such that*:

(P1) $\mathcal{I}$ has a solution if and only if so does $\mathcal{I}'$,
(P2) G' is acyclic and the graph $G'(P')$ is Eulerian, where $G'(P')$ is the graph obtained from G' after adding n edges from t to s for every $(s, t, n) \in P'$,
(P3) $|P'| \leq |P| + 1$ and $\mathbf{fvs}(\overline{G'}) \leq \mathbf{fvs}(\overline{G}) + 2$.

Proof The construction of $\mathcal{I}'$ is based on the construction of (G', H') from (G, H) in [28, Theorem 2]. Namely, for every $v \in V(G)$ let $\alpha(v) = \max\{0, \delta_{G(P)}(v) - \rho_{G(P)}(v)\}$ and $\beta(v) = \max\{0, \rho_{G(P)}(v) - \delta_{G(P)}(v)\}$ where $\delta_D(v)$ and $\rho_D(v)$ denote the number of out-neighbors respectively in-neighbors of a vertex v in a directed graph D. . Then

$$\begin{aligned} 0 =& |E(G(P))| - |E(G(P))| \\ =& \sum_{v \in V(G)} \left(\delta_{G(P)}(v) - \rho_{G(P)}(v)\right) \\ =& \sum_{v \in V(G)} (\alpha(v) - \beta(v)), \end{aligned}$$

where $G(P)$ is the graph obtained from G after adding n edges from t to s for every $(s, t, n) \in P$. Hence $\sum_{v \in V(G)} \alpha(v) = \sum_{v \in V(G)} \beta(v) = q$. We now construct the instance $\mathcal{I}' = (G', P')$ from (G, P) by adding one triple (s, t, q) to P as well as adding the vertices s and t together with $\alpha(v)$ edges from s to v and $\beta(v)$ edges from v to t for every $v \in V(G)$. It is straightforward to verify that $\mathcal{I}'$ satisfies Properties (P2) and (P3). Moreover, the reverse direction of Property (P1) is trivial. Towards showing the forward direction of Property (P1), assume that $\mathcal{I}$ has a solution S and let G'' be the graph obtained from G' after removing all edges appearing in a path in S. Then

$G''(\{(s,t,q)\})$ is Eulerian and G'' is acyclic, hence there are q pairwise edge-disjoint paths in G'' from s to t. □

We are now ready to show that MUEDP is W[1]-hard parameterized by the combined parameter the feedback vertex set number of the augmented graph and the number of terminal triples.

Lemma 10 *MUEDP is W[1]-hard parameterized by the following parameter: the feedback vertex set number of the augmented graph plus the number of terminal triples. Furthermore, this holds even when all integers on the input are given in unary.*

Proof We prove the lemma by a parameterized reduction from MDEDP. Namely, given an instance $\mathcal{I} = (G, P)$ of MDEDP we construct an equivalent instance $\mathcal{I}' = (H, Q)$ of MUEDP in polynomial time such that $|Q| \leq |P| + 1$, $\mathbf{fvs}(H^Q) \leq \mathbf{fvs}(G^P)$. The result then follows from Lemma 8.

Let (G', P') be the instance of MDEDP obtained from (G, P) by Lemma 9. Then $\mathcal{I}' = (H, P)$ is simply obtained from (G', P') by disregarding the directions of all edges in G'. Because of Lemma 9, we obtain that $|Q| \leq |P| + 1$ and $\mathbf{fvs}(H^Q) \leq \mathbf{fvs}(G^P)$ and it hence only remains to show the equivalence of $\mathcal{I}$ and $\mathcal{I}'$. Note that because of Lemma 9, it holds that (G', P') is acyclic and $G'(P')$ is Eulerian. It now follows from [28, Lemma 5] that (G', P') and (H, P) are equivalent, i.e., any solution for (G', P') is a solution for (H, P) and vice versa. Together with the equivalency between (G, P) and (G', P') (due to Lemma 9) this shows that $\mathcal{I}$ and $\mathcal{I}'$ are equivalent and concludes the proof of the lemma. □

Finally, using a very simple reduction from MUEDP we are ready to show that EDP is W[1]-hard parameterized by the feedback vertex set number of the augmented graph.

Theorem 3 EDP *is W[1]-hard parameterized by the feedback vertex set number of the augmented graph.*

Proof We prove the lemma by a parameterized reduction from MUEDP. Namely, given an instance $\mathcal{I} = (G, P)$ of MUEDP we construct an equivalent instance $\mathcal{I}' = (G', P')$ of EDP in polynomial time such that $\mathbf{fvs}(G'^{P'}) \leq \mathbf{fvs}(G^P) + 2|P|$. The result then follows from Lemma 10.

$\mathcal{I}'$ is obtained from $\mathcal{I}$ as follows. For every $(s,t,n) \in P$, we add $2n$ new vertices $s^1, \ldots, s^n$ and $t^1, \ldots, t^n$ to G and an edge between s^i and s as well as an edge between t^i and t for every $1 \leq i \leq n$. Moreover, for every i with $1 \leq i \leq n$, we add the terminal pair (s^i, t^i) to P'. It is straightforward to verify that $\mathcal{I}$ and $\mathcal{I}'$ are equivalent. Moreover, if F is a feedback vertex set for G^P, then $F \cup \{ s, t \mid (s,t,n) \in G \}$ is a feedback vertex set for $G'^{P'}$ and hence $\mathbf{fvs}(G'^{P'}) \leq \mathbf{fvs}(G^P) + 2|P|$, which concludes the proof of the theorem. □

6 An FPT-Algorithm for EDP Using the Augmented Graph

In light of Theorem 3, it is natural to ask whether there exist natural structural parameters of the augmented graph which would give rise to FPT-algorithms for EDP but which cannot be used on the input graph. In other words, does considering the augmented graph instead of the input graph provide any sort of advantage in terms of FPT-algorithms? In this section we answer this question affirmatively by showing that EDP is fixed-parameter tractable parameterized by the so-called *fracture number* of the augmented graph. We note that a parameter similar to the fracture number has recently been used to obtain FPT-algorithms for Integer Linear Programming [9].

Definition 5 A vertex subset X of a graph H is called a *fracture modulator* if each connected component in $H \setminus X$ contains at most $|X|$ vertices. We denote the size of a minimum-cardinality fracture modulator in H as $\text{frac}(H)$ or the *fracture number* of H.

Note the the fracture number is always at most the treewidth of the augmented graph and even though it is known that EDP parameterized by the treewidth of the augmented graph is in XP [29], it is open whether it is in FPT. Moreover, [12, Theorem 6] shows that EDP parameterized by the fracture number of the input graph is paraNP-hard.

Before we start our journey towards a fixed-parameter algorithm for EDP parameterized by the fracture number, we make a short digression related to the assumptions about EDP instances made in Sect. 2—notably, that each terminal occurs in a leaf of G and no two terminals occur on the same vertex. In particular, we observe that the assumption is "safe" to make when parameterizing by the fracture number.

Lemma 11 *There is a linear-time transformation which takes as input an arbitrary instance (G, P) of EDP and outputs an equivalent instance (G_0, P_0) of EDP such that $\text{frac}(G_0^{P_0}) \leq 3\text{frac}(G^P)^2 + \text{frac}(G^P)$ and (G_0, P_0) satisfies the following conditions: each vertex in a terminal pair has degree 1 and each vertex occurs in at most one terminal pair.*

Proof Consider an arbitrary fracture modulator X of G^P such that $|X| = \text{frac}(G^P) = k$ (which may, e.g., be computed using Lemma 13), and let (G_0, P_0) be the instance obtained from (G, P) by the reduction described in Sect. 2.2—in particular, for each $(a, b) \in P$, we create a new leaf a' adjacent to a and a new leaf b' adjacent to b, and replace (a, b) with (a', b'). Observe that each such newly created leaf of G_0 has degree 2 in $G_0^{P_0}$.

Consider an arbitrary vertex v in $V(G^P) - X$, and let C be the connected component of $G^P - X$ containing v. Note that v may only be adjacent to vertices in $X \cup C$ in G^P, and hence by the definition of the augmented graph we obtain that v occurs in at most $2k$ many sets in P. On the other hand, for a vertex $x \in X$ and a component

$C \in G - X$ there can be at most $|C| \leq k$ many tuples in P containing x and a vertex in C.

Now let us consider the graph $G_0^{P_0} - X$. For each component C' in $G_0^{P_0} - X$ such that there exists some $v \in C \cap V(G)$, let C be the component of $G^P - X$ containing v. Then C' consists of all vertices of C (at most k in total), all new leaf vertices created from these vertices in C (at most $2k^2$ in total), and all new leaf vertices created from vertices in X due to terminal pairs with one vertex in C (at most k^2 in total). As a consequence, $|C'| \leq 3k^2 + k$. On the other hand, if C' contains no vertex from $V(G)$ then it contains at most 2 newly created leaf vertices.

Since each component of $G_0^{P_0} - X$ has size at most $3k^2 + k$, it is easy to construct a fracture modulator of $G_0^{P_0}$ of that size: one may simply add an arbitrary set of $3k^2$ vertices to X. Hence $\mathrm{frac}(G_0^{P_0}) \leq 3k^2 + k$, as claimed. □

We now proceed with a simple structural observation about fracture modulators.

Lemma 12 *Let (G, P) be an instance of* EDP *and let k be the fracture number of its augmented graph. Then there exists a fracture modulator X of G^P of size at most $2k$ such that X does not contain any terminal vertices. Furthermore, such a fracture modulator X can be constructed from any fracture modulator of size at most k in linear time.*

Proof Let X' be arbitrary fracture modulator of size k and let $P' \subseteq P$ be the set of terminal pairs p such that $p \cap X' \neq \emptyset$. Consider the set $X = X' \setminus \widetilde{P'} \cup \{ N_G(a) \mid a \in \widetilde{P'} \}$. Because every terminal $a \in \widetilde{P}$ has at most one neighbor in G (recall our assumption on (G, P) given in Sect. 2.2), it holds that $|X| \leq 2|X'|$. Moreover, for vertex a that we removed from X', it holds that a is in a component of size two in $G^P \setminus X$, i.e., the component consisting of a and b where $\{a, b\} \in P'$. Consequently, every component of $G^P - X$ either has size two or it is a subset of a component of $G^P - X'$. □

We note that the problem of computing a fracture modulator of size at most k is closely related to the Vertex Integrity problem [8], and that a variant of it has been recently considered in the context of Integer Linear Programming [9].

Lemma 13 [9, Theorems 7 and 8] *There exists an algorithm which takes as input a graph G and an integer k, runs in time at most $\mathcal{O}((k+1)^k |E(G)|)$, and outputs a fracture modulator of cardinality at most k if such exists. Moreover, there is a polynomial-time algorithm that either computes a fracture modulator of size at most $(k+1)k$ or outputs correctly that no fracture modulator of size at most k exists.*

Proof The algorithm is based on a bounded search tree approach and relies on the following observations.

(O1) If G is not connected then each fracture modulator of G is the disjoint union of fracture modulators for all connected components of G.

(O2) If G is connected and D is any set of $k+1$ vertices of G such that $G[D]$ is connected, then any fracture modulator has to contain at least one vertex from D.

These observations lead directly to the following recursive algorithm that either determines that the instance (G, k) is a NO-instance or outputs a fracture modulator X of size at most k. The algorithm also remembers the maximum size of any component in a global constant c, which is set to k for the whole duration of the algorithm. The algorithm first checks whether G is connected. If G is not connected the algorithm calls itself recursively on the instance $(G[C], k)$ for each connected component C of G. If one of the recursive calls returns NO or if the size of the union of the solutions returned for each component exceeds k, the algorithm returns that I is a NO-instance. Otherwise the algorithm returns the union of the solutions returned for each component of G.

If G is connected and $|V(G)| \leq c$, the algorithm returns the empty set as a solution. Otherwise, i.e., if G is connected but $|V(G)| > c$ the algorithm first computes a set D of $c+1$ vertices of G such that $G[D]$ is connected. This can for instance be achieved by a depth-first search that starts at any vertex of G and stops as soon as $c+1$ vertices have been visited. The algorithm then branches on the vertices in D, i.e., for every $v \in D$ the algorithm recursively computes a solution for the instance $(G - \{v\}, k-1)$. It then returns the solution of minimum size returned by any of those recursive calls, or NO if none of those calls return a solution. This completes the description of the algorithm. The correctness of the algorithm follows immediately from the above observations. Moreover the running time of the algorithm is easily seen to be dominated by the maximum time required for the case that at each step of the algorithm G is connected.

In this case the running time can be obtained as the product of the number of branching steps times the time spent on each of those. Because at each recursive call the parameter k is decreased by at least one and the number of branching choices is at most $c+1$, we obtain that there are at most $(c+1)^k = (k+1)^k$ branching steps. Furthermore, the time at each branching step is dominated by the time required to check whether G is connected, which is linear in the number of edges of G. Putting everything together, we obtain $\mathcal{O}((k+1)^k|E(G)|)$ as the total time required by the algorithm, which completes the proof of the lemma. □

We note that the depth-first search algorithm in the above proof can be easily transformed into a polynomial time approximation algorithm for finding fracture modulators, with an approximation ratio of $k+1$. In particular, instead of branching on the vertices of a connected subgraph D of G with $k+1$ vertices, this algorithm would simply add all the vertices of D into the current solution.

For the rest of this section, let us fix an instance (G, P) of EDP with a fracture modulator X of G^P of cardinality k which does not contain any terminals. Furthermore, since the subdivision of any edge (i.e., replacing an edge by a path of length 2) in (G, P) does not change the validity of the instance, we will assume without loss of generality that $G[X]$ is edgeless; in particular, any edges that may have had both endpoints in X will be subdivided, creating a new connected component of size 1.

Our next step is the definition of *configurations*. These capture one specific way a connected component C of $G^P - X$ may interact with the rest of the instance. It will be useful to observe that for each terminal pair ab there exists precisely one connected component C of $G^P - X$ which contains both of its terminals; we say that ab *occurs* in C. For a connected component C, we let C^+ denote the induced subgraph on $G^P[C \cup X]$.

A *trace* is a tuple $(x_1, \ldots, x_\ell)$ of elements of X. A configuration is a tuple (α, β) where

- α is a multiset of at most k traces, and
- β is a mapping from subsets of X of cardinality 2 to $[k^2]$; here k^2 is an upperbound on the number of edge-disjoint paths between two vertices in X using only edges in some component.

A component C of G^P *admits* a configuration (α, β) if there exists a set of edge disjoint paths $\mathcal{F}$ in C^+ and a surjective mapping τ (called the *assignment*) from α to the terminal pairs that occur in C with the following properties.

- For each terminal pair st that occurs in C:
  - $\mathcal{F}$ either contains a path whose endpoints are s and t, or
  - $\mathcal{F}$ contains an s-x_1 path for some $x_1 \in X$ and a t-x_2 path for some distinct $x_2 \in X$ and there exists a trace $L = (x_1, \ldots, x_2) \in \alpha$ such that $\tau(L) = st$.
- for each distinct $a, b \in X$, $\mathcal{F}$ contains precisely $\beta(\{a, b\})$ paths from a to b.
- $\mathcal{F}$ contains no other paths than the above.

Intuitively, α stores information about how one particular set of edge disjoint paths A which originate in C is routed through the instance: they may either be routed only through C^+ (in which case they don't contribute to α), or they may leave C^+ (in which case α stores the order in which these paths visit vertices of X, i.e., their trace). On the other hand, β stores information about how paths that originate outside of C can potentially be routed through C (in a way which does not interfere with A). Observe that for any particular α there may exist several distinct configurations $((\alpha, \beta_1), (\alpha, \beta_2)$ and so forth); similarly, for any particular β there may exist several distinct configurations $((\alpha_1, \beta), (\alpha_2, \beta)$ and so forth).

If a set $\mathcal{F}$ of edge disjoint paths in C^+ satisfies the conditions specified above for a configuration (α, β), we say that $\mathcal{F}$ *gives rise* to (α, β). Clearly, given $\mathcal{F}$ and (α, β), it is possible to determine whether $\mathcal{F}$ gives rise to (α, β) in time polynomial in $|V(C)|$.

While configurations capture information about how a component can interact with a set of edge disjoint paths, our end goal is to have a way of capturing all important information about a component irrespective of any particular selection of edge disjoint paths. To this end, we introduce the notion of *signatures*. A signature of a component C, denoted $\text{sign}(C)$, is the set of all configurations which C admits. The set of all configurations is denoted by Λ.

Lemma 14 *Given a component C, it is possible to compute* $\text{sign}(C)$ *in time at most* $k^{\mathcal{O}(k^2)}$. *Furthermore*, $|\text{sign}(C)| \leq |\Lambda| \leq k^{\mathcal{O}(k^2)}$.

Proof We begin with the latter claim. The number of traces is $k! + (k-1)! + \cdots + 1!$, which is upper-bounded by $2 \cdot k!$. Consequently, the number of choices for α is upper-bounded by $(2 \cdot k!)^k \leq k^{\mathcal{O}(k^2)}$. On the other hand, the number of choices for β is upper-bounded by $(k^2)^{k^2}$. Since the number of configurations is upper-bounded by the number of choices for α times the number of choices for β, we obtain $|\Lambda| \leq k^{\mathcal{O}(k^2)}$. Note that it is possible to exhaustively construct Λ in the same time bound.

For the former claim, observe that C^+ contains at most $2k^2$ edges. Consider the following exhaustive algorithm on C^+. The algorithm exhaustively tries all assignments of edges in C^+ to labels from $\{\emptyset\} \cup [2k^2]$. For each such assignment$\mathcal{F}$, it checks that each label forms a path in C^+; if this is the case, then $\mathcal{F}$ is a collection of edge disjoint paths of C^+. We can then loop through each configuration (α, β) in Λ, check whether $\mathcal{F}$ gives rise to (α, β), and if yes we add (α, β) to $\text{sign}(C)$. In total, this takes time at most $k^{\mathcal{O}(k^2)} \cdot k^{\mathcal{O}(k^2)} = k^{\mathcal{O}(k^2)}$. □

Our next step is the formulation of a clear condition linking configurations of components in $G^P - X$ and solving (G, P). This condition will be of importance later, since it will be checkable by an integer linear program. For a trace α, we say that a, b *occur consecutively* in α if elements a and b occur consecutively in the sequence α (regardless of their order), i.e., $\alpha = (\ldots, a, b, \ldots)$ or $\alpha = (\ldots, b, a, \ldots)$. Let $\mathcal{D}$ be the set of connected components of $G^P - X$.

A *configuration selector* is a function which maps each connected component C in $G^P - X$ to a configuration $(\alpha, \beta) \in \text{sign}(C)$. We say that a configuration selector S is *valid* if it satisfies the condition that $\text{dem}(ab) \leq \text{sup}(ab)$ for every $\{a, b\} \subseteq X$, where dem (*demand*) and sup (*supply*) are defined as follows:

- $\text{dem}(ab)$ is the number of traces in $\bigcup_{C \in \mathcal{D}} S(C)$ where ab occur consecutively.
- $\text{sup}(ab)$ is the sum of all the values $\beta(a, b)$ in $\bigcup_{C \in \mathcal{D}} S(C)$.

For completeness, we provide the formal definitions of $\text{sup}(ab)$ and $\text{dem}(ab)$ below.

- $\text{dem}(ab)$: let the multiset $\mathcal{A}^0$ be the restriction of the multiset $\bigcup_{C \in \mathcal{D}} S(C)$ to the first component of each configuration. Then we set $\mathcal{A} = \bigcup_{\alpha \in \mathcal{A}^0} \alpha$, i.e., $\mathcal{A}$ is the multiset of all traces which occur in configurations originating from S. Finally, let $\mathcal{A}_{ab}$ be the restriction of $\mathcal{A}$ only to those traces where ab occurs consecutively, and we set $\text{dem}(ab) = |\mathcal{A}_{ab}|$.
- $\text{sup}(ab)$: Stated formally, let the multiset $\mathcal{B}^0$ be the restriction of the multiset $\bigcup_{C \in \mathcal{D}} S(C)$ to the second component of each configuration. Then we set $\mathcal{B} = \bigcup_{\beta \in \mathcal{B}^0} \beta$, i.e., $\mathcal{B}$ is the multiset of all mappings which occur in configurations originating from S. Finally, we set $\text{sup}(ab) = \sum_{\beta \in \mathcal{B}} \beta(ab)$.

The next, crucial lemma links the existence of a valid configuration selector to the existence of a solution for EDP.

Lemma 15 *(G, P) is a yes-instance if and only if there is a valid configuration selector.*

Proof Assume (G, P) is a yes-instance and let Q be a solution, i.e., Q is a set of edge disjoint paths in G which link each terminal pair in P. We will construct a valid configuration selector S. First, consider a connected component C of $G^P - X$. Observe that Q restricted to C^+ forms a set of edge disjoint paths Q_C which consists of:

- paths starting and ending in X,
- paths starting and ending at terminals of a terminal pair, and
- paths starting at terminals of a terminal pair and ending in X.

We will use Q_C to construct a configuration (α_C, β_C) of C, as follows. For each $\{x, y\} \subseteq X$, $\beta_C(\{x, y\})$ will map each tuple $\{x, y\}$ to the number of paths in Q_C whose endpoints are precisely x and y. On the other hand, for each pair of paths in Q_C which start at terminals s, t of a terminal pair and end at $x, y \in X$, we add the trace $(x, z_1, \ldots, z_\ell, y)$ to α, where z_i is the $(i + 1)$-th vertex visited by the s-t path of Q in X. For example, if the s-t path used by the solution intersects with vertices in X in the order (a, b, c, d), then we add the trace (a, b, c, d) to α_C. As witnessed by Q_C, the resulting configuration (α_C, β_C) is a configuration of C and in particular $(\alpha_C, \beta_C) \in \text{sign}(C)$.

Let S be a configuration selector which maps each connected component C to (α_C, β_C) constructed as above. We claim that S is valid. Indeed, for each $\{a, b\} \in X$, $\text{dem}(ab)$ is the number of terminal-to-terminal paths in Q which consecutively visit a and then b in X (or vice-versa, b and then a). At the same time, $\sup(ab)$ is the number of path segments in Q whose endpoints are a and b. Clearly, $\text{dem}(ab) = \sup(ab)$.

On the other hand, assume we have a valid configuration selector S. We will use S to argue the existence of a set Q of edge disjoint paths which connect all terminal pairs in (G, P). For each component C we know that C admits $S(C) = (\alpha_C, \beta_C)$ and hence there exists a set of edge disjoint paths, say $\mathcal{F}_C$, which satisfies the following conditions:

- For each terminal pair st in C, $\mathcal{F}_C$ either contains a path whose endpoints are s and t or two paths from s and t to two distinct endpoints in X;
- For each distinct $a, b \in X$, $\mathcal{F}_C$ contains precisely $\beta_C(\{a, b\})$ paths from a to b.

Let $\mathcal{F} = \bigcup_{C \in \mathcal{D}} \mathcal{F}_C$, and let τ_C be the surjective assignment function for C which goes with $\mathcal{F}_C$. We will now construct the set Q from $\mathcal{F}$ as follows. For each terminal pair s, t in some component C, we add a s-t path L into Q, where L is obtained as follows. If $\mathcal{F}_C$ contains a path whose endpoints are s and t, then we set this to L. Otherwise, L is composed of the following segments:

- the two paths in $\mathcal{F}_C$ whose endpoints are s and t, and
- for each $x, y \in X$ which occur consecutively in the trace $\tau_C^{-1}(st)$, we choose an arbitrary x-y path segment from $\mathcal{F}$, use it for L, and then delete this path segment from $\mathcal{F}$.

First of all, observe that since S is valid, there will always be enough x-y path segments in $\mathcal{F}$ to choose from in the construction of L. Furthermore, each L is an s-t path, and all the paths in Q are edge disjoint. Hence (G, P) is indeed a yes-instance and the lemma holds. □

Lemma 16 *There exists an algorithm which takes as input an* EDP *instance* (G, P) *and a fracture modulator* X *of* G^P *and determines whether there exists a valid configuration selector* S *in time at most* $2^{2^{k^{\mathcal{O}(k^2)}}} \cdot |V(G)|$.

Proof Consider the following instance of Integer Linear Programming. For each signature η such that there exists a connected component C of $G^P - X$ where $\text{sign}(C) = \eta$, and for each configuration $(\alpha, \beta) \in \eta$, we create a variable $z^{\eta}_{(\alpha,\beta)}$, and for each we add a constraint requiring it to be nonnegative. The first set of constraints we create is as follows: for each signature η, we set

$$\sum_{(\alpha,\beta)\in\eta} z^{\eta}_{(\alpha,\beta)} = d_{\eta},$$

where d_η is the number of connected components of $G^P - X$ with signature η.

Next, for each $\{x, y\} \subseteq X$ and for $i \in [k]$, let $Y_i^{x,y}$ be the set of all configurations (α, β) such that the number of traces in α where x, y occur consecutively is i. Similarly, for $i \in [k^2]$, let $\Lambda_i^{x,y}$ be the set of all configurations (α, β) such that $\beta(\{x, y\}) = i$. These sets are used to aggregate all configurations with a specific "contribution" i to the demand and supply. For example, all configurations whose first component is $\alpha = \{(a, b, c), (a, d, e, b), (e, b, a)\}$ would belong to $Y_2^{a,b}$, and similarly all configurations whose second component β satisfy $\beta(\{a, b\}) = 3$ would belong to $\Lambda_3^{x,y}$. We can now add our second set of constraints: for each $\{x, y\} \subseteq X$,

$$\sum_{i\in[k]} \left(i \cdot \sum_{(\alpha,\beta)\in Y_i^{x,y}} z^{\eta}_{(\alpha,\beta)} \right) \leq \sum_{i\in[k^2]} \left(i \cdot \sum_{(\alpha,\beta)\in \Lambda_i^{x,y}} z^{\eta}_{(\alpha,\beta)} \right).$$

The number of variables used in the ILP instance is upper-bounded by the number of signatures times the cardinality of the largest signature. By Lemma 14, the latter is upper-bounded by $k^{\mathcal{O}(k^2)}$, and therefore the former is upper-bounded by $2^{k^{\mathcal{O}(k^2)}}$; in total, the instance thus contains at most $2^{k^{\mathcal{O}(k^2)}}$ variables. Since we can determine $\text{sign}(C)$ for each connected component C of $G^P - X$ by Lemma 14 and the number of connected components is upper-bounded by $|V(G)|$, we can also construct this ILP instance in time at most $2^{k^{\mathcal{O}(k^2)}} \cdot |V(G)|$. Moreover, Proposition 1 allows us to solve the ILP instance in time at most $2^{2^{k^{\mathcal{O}(k^2)}}}$, and in particular to output an assignment ζ from variables $z^{\eta}_{(\alpha,\beta)}$ to integers which satisfies the above constraints.

To conclude the proof, we show how ζ can be used to obtain the desired configuration selector S. For each set of connected components with signature η, let S map precisely $z^{\eta}_{(\alpha,\beta)}$ connected components to configuration (α, β). Observe that due to the first set of constraints, in total S will be mapping precisely the correct number of components with signature η to individual configurations (α, β). Moreover, the

second set of constraints ensures that, for every $\{x, y\} \subseteq X$, the total demand does not exceed the total supply. Hence S is valid and the lemma holds. □

Theorem 4 EDP is fixed-parameter tractable parameterized by the fracture number of the augmented graph.

Proof We begin by computing a fracture modulator of the augmented graph by Lemma 13. We then use Lemma 16 to determine whether a valid configuration selector S exists, which by Lemma 15 allows us to solve EDP. □

7 Hardness for Large Components with One Terminal Pair

In the previous section we have shown that EDP is fixed-parameter tractable parameterized by the fracture number. One might be tempted to think that tractability still applies if instead of bounding the size of each component one only bounds the number of terminal pairs in each component. In this section we show that this is not the case, i.e., we show that even if both the deletion set as well as the number of terminal pairs in each component are bounded by a constant, EDP remains NP-complete. Namely, this section is devoted to a proof of the following theorem.

Theorem 5 EDP *is* NP*-complete even if the augmented graph G^P of the instance has a deletion set D of size 6 such that each component of $G^P - D$ contains at most 1 terminal pair*">.

We will show the theorem by a polynomial-time reduction from the following problem.

Multiple edge disjoint Paths (MEDP)	
Input:	An undirected graph G, three pairs (s_1, t_1), (s_2, t_2), and (s_3, t_3) of terminals (vertices of G) and three integers n_1, n_2, and n_3.
Question:	Is there a set of pairwise edge disjoint paths containing n_1 paths between s_1 and t_1, n_2 paths between s_2 and t_2, and n_3 paths between s_3 and t_3?

It is shown in [28, Theorem 4] that MEDP is strongly NP-complete.

Proof of Theorem 5 We provide a polynomial-time reduction from the MEDP problem. Namely, for an instance $\mathcal{I} = (G, (s_1, t_1), (s_2, t_2), (s_3, t_3), n_1, n_2, n_3)$ of MEDP, we construct an instance $\mathcal{I}' = (H, P)$ in polynomial time such that $\mathcal{I}$ has a solution if and only if so does $\mathcal{I}'$ and H^P has a deletion set $D \subseteq V(H)$ of size 6 such that each component of $H^P - D$ contains at most one terminal pair.

The graph H is obtained from G after adding:

(C1) for every i with $1 \leq i \leq n_1$ two new vertices s_1^i and t_1^i, where s_1^i is connected by an edge to s_1 and t_1^i is connected by an edge to t_1,
(C2) for every i with $1 \leq i \leq n_2$ two new vertices s_2^i and t_2^i, where s_2^i is connected by an edge to s_2 and t_2^i is connected by an edge to t_2.
(C3) for every i with $1 \leq i \leq n_3$ two new vertices s_3^i and t_3^i, where s_3^i is connected by an edge to s_3 and t_3^i is connected by an edge to t_3.

This completes the description of H. The set P is defined as $\{ (s_1^i, t_1^i) \mid 1 \leq i \leq n_1 \} \cup \{ (s_2^i, t_2^i) \mid 1 \leq i \leq n_2 \} \cup \{ (s_3^i, t_3^i) \mid 1 \leq i \leq n_3 \}$. Observe that after setting the deletion set D to $\{s_1, t_1, s_2, t_2, s_3, t_4\}$, the graph $H^P - D$ has one component consisting of the vertices s_1^i and t_1^i for every i with $1 \leq i \leq n_1$, one component consisting of the vertices s_2^i and t_2^i for every i with $1 \leq i \leq n_2$, one component consisting of the vertices s_3^i and t_3^i for every i with $1 \leq i \leq n_3$, and the possibly large components of $G - D$. Hence every component of $H^P - D$ contains at most one terminal pair.

It remains to show that $\mathcal{I}$ has a solution if and only if so does $\mathcal{I}'$. For the forward direction let S be a solution for $\mathcal{I}$, i.e., S is a set of pairwise edge disjoint paths in G containing n_1 paths $P_1, \ldots, P_{n_1}$ between s_1 and t_1, n_2 paths $Q_1, \ldots, Q_{n_2}$ between s_2 and t_2, and n_3 paths $R_1, \ldots, R_{n_2}$ between s_3 and t_3. Then we obtain a solution for $\mathcal{I}$, i.e., a set S' of pairwise edge disjoint paths in H containing one path between every terminal pair in P as follows. For every i with $1 \leq i \leq n_1$, we add the path between s_1^i and t_1^i in H defined as $(s_1^i, s_1) \circ P_i \circ (t_1, t_1^i)$ to S', where $S_1 \circ S_2$ denotes the concatenation of the two sequences S_1 and S_2. Similarly, for every i with $1 \leq i \leq n_2$, we add the path between s_2^i and t_2^i in H defined as $(s_2^i, s_2) \circ Q_i \circ (t_2, t_2^i)$ to S'. Finally, for every i with $1 \leq i \leq n_3$, we add the path between s_3^i and t_3^i in H defined as $(s_3^i, s_3) \circ R_i \circ (t_3, t_3^i)$ to S'. Because all the path added to S' are pairwise edge disjoint and we added a path for every terminal pair in P, this shows that S' is a solution for $\mathcal{I}'$.

Towards showing the reverse direction, let S' be a solution for $\mathcal{I}'$, i.e., a set of pairwise edge disjoint paths containing one path, denoted by P_p for every terminal pair $p \in P$. Observe that if $p = (s_1^i, t_1^i)$ for some i with $1 \leq i \leq n_1$, then P_p contains a path between s_1 and t_1 in G. The same applies if $p = (s_2^i, t_2^i)$ for some i with $1 \leq i \leq n_2$ and $p = (s_3^i, t_3^i)$ for some i with $1 \leq i \leq n_3$. Hence the set S containing the restriction of every path P_p in S' to G contains n_1 paths between s_1 and t_1, n_2 paths between s_2 and t_2 in G, and n_3 paths between s_3 and t_3, which are all pairwise edge disjoint. Hence S is a solution for $\mathcal{I}$. □

8 Conclusion

Our results close a wide gap in the understanding of the complexity landscape of EDP parameterized by structural parameterizations. On the positive side we present three novel algorithms for the classical EDP problem: a polynomial-time algorithm for instances with a FVS of size one, an FPT-algorithm w.r.t. the treewidth and maximum degree of the input graph, and an FPT-algorithm for instances that

have a small deletion set into small components in the augmented graph. On the negative side we solve a long-standing open problem concerning the complexity of EDP parameterized by the treewidth of the augmented graph: unlike related multicut problems [18], EDP is W[1]-hard parameterized by the feedback vertex set number of the augmented graph.

Acknowledgements Robert Ganian acknowledges support by the Austrian Science Fund (FWF), Projects P31336 and Y1329. Sebastian Ordyniak acknowledges support from the Engineering and Physical Sciences Research Council (EPSRC, project EP/V00252X/1). The authors also wish to thank the anonymous reviewers for their exceptionally detailed and helpful comments.

References

1. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM J. Comput. **25**(6), 1305–1317 (1996)
2. Bodlaender, H.L., Grønås Drange, P., Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A c^k n 5-approximation algorithm for treewidth. SIAM J. Comput., 45(2):317–378, (2016)
3. Bodlaender, H.L., Kloks, T.: Efficient and constructive algorithms for the pathwidth and treewidth of graphs. J. Algorithms **21**(2), 358–402 (1996)
4. Chekuri, C., Khanna, S., Shepherd, F.B.: An O(sqrt(n)) approximation and integrality gap for disjoint paths and unsplittable flow. Theory Comput. **2**(7), 137–146 (2006)
5. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D.: Dániel Marx. Michal Pilipczuk, and Saket Saurabh. Parameterized Algorithms. Springer, Marcin Pilipczuk (2015)
6. Diestel, R.: Graph Theory, 4th edn. Springer, Heidelberg (2010)
7. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science. Springer, Berlin (2013)
8. Grønås Drange, P., Dregi, M.S., van 't Hof, P, : On the computational complexity of vertex integrity and component order connectivity. Algorithmica **76**(4), 1181–1202 (2016)
9. Dvořák, P., Eiben, E., Ganian, R., Knop, D., Ordyniak, S. Solving integer linear programs with a small number of global variables and constraints. In: Proceedings of the IJCAI 2017, 2017 (to appear)
10. Ene, A., Mnich, M., Pilipczuk, M., Risteski, A. On routing disjoint paths in bounded treewidth graphs. In: Proceedings of the SWAT 2016, volume 53 of LIPIcs, pp. 15:1–15:15. Schloss Dagstuhl (2016)
11. Fellows, M.R., Lokshtanov, D., Misra, N., Rosamond, F.A., Saurabh, S. Graph layout problems parameterized by vertex cover. In: ISAAC, Lecture Notes in Computer Science, pp. 294–305. Springer (2008)
12. Fleszar, K., Mnich, M., Spoerhase, J. New algorithms for maximum disjoint paths based on tree-likeness. In: Proceedings of the ESA 2016, pp. 42:1–42:17 (2016)
13. Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science, vol. XIV. An EATCS Series. Springer, Berlin (2006)
14. Frank, A., Tardos, É.: An application of simultaneous diophantine approximation in combinatorial optimization. Combinatorica **7**(1), 49–65 (1987)

15. Ganian, R., Klute, F., Ordyniak, S. On structural parameterizations of the bounded-degree vertex deletion problem. Algorithmica (to appear), 2000. to appear. https://doi.org/10.1007/s00453-020-00758-8
16. Ganian, R., Ordyniak, S. The power of cut-based parameters for computing edge disjoint paths. Algorithmica, 2020 (to appear)
17. Garg, N., Vazirani, V.V., Yannakakis, M.: Primal-dual approximation algorithms for integral flow and multicut in trees. Algorithmica **18**(1), 3–20 (1997)
18. Georg Gottlob and Stephanie Tien Lee: A logical approach to multicut problems. Inf. Process. Lett. **103**(4), 136–141 (2007)
19. Kannan, R.: Minkowski's convex body theorem and integer programming. Math. Oper. Res. **12**(3), 415–440 (1987)
20. Karp, RiM: On the computational complexity of combinatorial problems. Networks **5**(1), 45–68 (1975)
21. Ken-ichi, K., Kobayashi, Y., Kreutzer, S. An excluded half-integral grid theorem for digraphs and the directed disjoint paths problem. In Proc. STOC 2014, pp. 70–78. ACM (2014)
22. Kloks, T.: Treewidth: Computations and Approximations. Springer, Berlin (1994)
23. Kolliopoulos, S.G., Stein, C.: Approximating disjoint-path problems using packing integer programs. Math. Program. **99**(1), 63–87 (2004)
24. Lenstra, H.W.: Integer programming with a fixed number of variables. Math. Oper. Res, 8(4):538–548 (1983)
25. Micali, S., Vazirani, V.V. An O(sqrt(|V|)|E|) algorithm for finding maximum matching in general graphs. In 21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13-15 October 1980, pp. 17–27. IEEE Computer Society, 1980
26. Nishizeki, T., Vygen, J., Zhou, X.: The edge-disjoint paths problem is NP-complete for series-parallel graphs. Discrete Appl. Math. **115**(1–3), 177–186 (2001)
27. Robertson, N., Seymour, P.D.: Graph minors xiii the disjoint paths problem. J. Comb. Theory Ser. B **63**(1), 65–110 (1995)
28. Vygen, J.: Np-completeness of some edge-disjoint paths problems. Discrete Appl. Math. **61**(1), 83–90 (1995)
29. Zhou, X., Tamura, S., Nishizeki, T.: Finding edge-disjoint paths in partial k-trees. Algorithmica **26**(1), 3–30 (2000)