Algorithmica (2023) 85:1202–1250
https://doi.org/10.1007/s00453-022-01064-1

The Complexity of Routing Problems in Forbidden-Transition Graphs and Edge-Colored Graphs

Thomas Bellitto[1,2] · Shaohua Li[2] · Karolina Okrasa[2,3] · Marcin Pilipczuk[2] · Manuel Sorge[2]

Received: 2 October 2021 / Accepted: 10 November 2022 / Published online: 4 December 2022

Abstract

The notion of *forbidden-transition graphs* allows for a robust generalization of walks in graphs. In a forbidden-transition graph, every pair of edges incident to a common vertex is *permitted* or *forbidden*; a walk is *compatible* if all pairs of consecutive edges on the walk are permitted. Forbidden-transition graphs and related models have found applications in a variety of fields, such as routing in optical telecommunication networks, road networks, and bio-informatics. A widely-studied special case are edge-colored graphs, where a compatible walk is forbidden to take two edges of the same color in a row. We initiate the study of fundamental problems on finding paths, cycles and walks in forbidden-transition graphs from the point of view of parameterized complexity, including an in-depth study of tractability with regards to various graph-width parameters. Among several results, we prove that finding a simple compatible path between given endpoints in a forbidden-transition graph is $W[1]$-hard

This research is a part of a project that have received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme Grant Agreement 714704. Parts of Manuel Sorge's work were performed while visiting TU Wien, Vienna, Austria.

✉ Karolina Okrasa
k.okrasa@mini.pw.edu.pl

Thomas Bellitto
thomas.bellitto@lip6.fr

Shaohua Li
S.Li@mimuw.edu.pl

Marcin Pilipczuk
malcin@mimuw.edu.pl

Manuel Sorge
m.sorge@uw.edu.pl

1 Sorbonne Université, CNRS, LIP6, Paris, France

2 Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Warsaw, Poland

3 Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw, Poland

when parameterized by the vertex-deletion distance to a linear forest (so it is also hard when parameterized by pathwidth or treewidth). On the other hand, we show an algebraic trick that yields tractability when parameterized by treewidth for finding a compatible Hamiltonian cycle in the edge-colored graph setting.

Keywords Graph algorithms · Fixed-parameter tractability · Parameterized complexity

1 Introduction

Graphs have proved to be an extremely useful tool to model routing problems in a very wide range of applications. However, we sometimes need to express constraints on the permitted walks that are stronger than what the standard graph model allows for. For example, in a road network, there can be a crossroad where drivers are not allowed to turn right. In this case, many walks in the underlying graph without transition restrictions would correspond to routes that a driver is not allowed to use. To overcome this limitation, Kotzig introduced forbidden-transition graphs [39]. Let G be an undirected graph. A *transition* in G is an unordered pair of adjacent edges. Every time a walk in G uses two edges uv and vw consecutively, we say that the walk *uses the transition* $\{uv, vw\}$. A *transition system* of G is a set of transitions in G. A *forbidden-transition graph* is a tuple (G, T) of a graph G together with a transition system T of G.[1] We say that a transition is *permitted* if it is in T and it is *forbidden* otherwise. We say a walk is *compatible* with T, or T*-compatible*, if all the transitions it uses are permitted, that is, they are in T. We omit reference to T when it is clear from the context. For notational clarity, it is sometimes useful to refer to the transitions $T(v)$ of a specific vertex $v \in V(G)$, that is, $T(v) = \{\{e, f\} \in T \mid e \cap f = \{v\}\}$.

Since their introduction, forbidden-transition graphs and related models have found applications in a variety of fields, such as routing in optical telecommunication networks [2], road networks [6], and bio-informatics [16]. Problems of routing, connectivity, and robustness in those graphs have received a lot of attention but unfortunately, those problems generally turn out to be algorithmically very difficult, even on very restricted subclasses of graphs. In [49], Szeider famously proved that even determining the existence of a compatible (elementary) path between two given vertices of a forbidden-transition graph is NP-complete. Similarly, many known results about forbidden-transition graphs are proofs of NP-completeness of problems that are polynomially solvable on standard graphs (e.g. [1], [7], [18], [27], [28], [34], [35], [49]).

A very interesting specific case of compatible walks in forbidden-transition graphs are properly colored walks in edge-colored graphs. Here, a graph is given together with a coloring of its edges and we say that a walk is *properly colored* if it does not use consecutively two edges of the same color. These graphs have been introduced by Dorninger in [16] to study chromosome arrangements. The problem of properly colored Hamiltonian cycles was the first problem explicitly studied on edge-colored

[1] Our notation rather suggests that (G, T) is a *permitted-transition graph* but we use forbidden transitions in keeping with convention in the literature.

graphs and this problem and its variants (such as longest elementary cycle or spanning trails among many others) are especially well studied in the literature. We refer the reader to [30] or [5] for surveys on these problems and to [4], [13], [14], [31], [40] or [41] for recent developments.

We remark that edge-colored graphs with two colors generalize directed graphs in the following sense. Assume that the colors are red and blue, take a directed graph G, and replace every arc $e = (u, v) \in E(G)$ with a new vertex x_e, a blue edge ux_e and a red edge vx_e. One can observe that each properly colored walk in the resulting graph corresponds to a walk in G' that respects arc directions (but may go backwards). Using this observation, for most of the problems asking for paths, cycles, and walks, it is easy to provide a reduction from the directed graph setting to an edge-colored setting with two colors.

Because of their expressiveness and wide range of applications, the study of forbidden-transition graphs is a fast-emerging field and has been the subject of growing attention in the past decades but we are still very far from understanding them as well as regular graphs. Our aim in this paper is to study the parameterized complexity of some known NP-complete problems concerning existence of paths, cycles, or walks (which is the natural class of problems in this context), in general forbidden-transition graphs as well as in the specific case of edge-colored graphs. We specifically focus on some problems of great practical interest, such as the existence of an elementary path or the length of a shortest path between given vertices, the problem of Hamiltonian cycles, or linkage problems where we try to connect pairs of vertices by vertex- or edge-disjoint paths. A very rich toolbox already exists to study fixed-parameter tractability in standard graphs (see [15] for example) but the generalization of these concepts to forbidden-transition graphs is widely unexplored and raises many challenges that we hope to see get more attention in the future.

1.1 Our Results

In Sect. 3, we study the problem of shortest compatible paths between two vertices s and t in a forbidden-transition graph. Recall that determining whether there exists a compatible path between s and t is known to be NP-complete [49]. A simple application of the color-coding technique shows that this problem is fixed-parameter tractable when parameterized by the length of the path. We improve upon this observation by showing that the complexity of finding a shortest compatible path from s to t is actually fixed-parameter tractable when parameterized by the length of the detour that the forbidden transitions impose. In other words, determining whether there exists a compatible path of length at most $d(s, t) + k$ where $d(s, t)$ is the length of the shortest path between s and t in the underlying graph with no forbidden transitions, is fixed-parameter tractable when parameterized by k. Our algorithm follows the main ideas of the algorithm for the Exact Detour problem by Bezáková et al. [10].

In Sect. 4, we turn our attention to graph-width parameters. The rich ecosystem of relevant graph-width parameters is depicted in Fig. 1; see [24, 44, 51] for the corresponding boundedness and unboundedness relations on treecut-width.

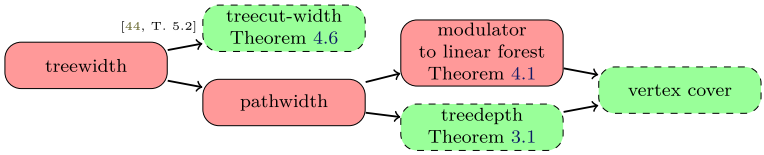

Fig. 1 A hierarchy of graph-width parameters considered in this work. An arrow from a to b represents the fact that a bound on parameter b imposes a bound on parameter a, but there exist families of graphs with bounded a and unbounded b. A parameter a is green and dashed if detecting a compatible s-t path is fixed-parameter tractable with respect to a; the parameter is red and solid if this problem is W[1]-hard (Color figure online)

First, we focus on the NP-complete problem of determining whether there exists a compatible path between s and t in a forbidden-transition graph. Since the problem is fixed-parameter tractable when parameterized by the length of the path (see Sect. 3), it is also fixed-parameter tractable when parameterized by the vertex cover number or the treedepth of the graph, as bounding the vertex cover number or the treedepth of the graph by k bounds the length of the longest simple path by $2k$ or $2^k - 1$, respectively. Our main result is a negative one: the problem becomes $W[1]$-hard if one makes one step further to the parameter *modulator to a linear forest*, i.e., the number of vertices one has to remove from the graph to turn it into a union of vertex-disjoint paths. A small tweak of the reduction shows that finding a Hamiltonian cycle is $W[1]$-hard with respect to the size of a modulator to treewidth 2. Our reduction in particular implies hardness for the parameters pathwidth and treewidth (for both the compatible path and Hamiltonian cycle problems). The reduction is technically involved and is the main negative result in this paper.

On the other hand, we show that if one considers parameters based on edge cuts (as opposed to vertex cuts, like in treewidth), one can obtain nontrivial tractability results. *Treecut-width* is a width notion based on edge cuts, introduced by Wollan [51], and playing the role of treewidth in the world of the immersion relation. We prove that the problem of finding a compatible s-t path is fixed-parameter tractable when parameterized by the treecut-width of the graph. More precisely, the problem can be solved in time $k^{\mathcal{O}}(k^2) \cdot n^3$ where k denotes the treecut-width.

In the light of the hardness in general forbidden-transition graphs of detecting s-t paths, the most fundamental connectivity problem, we move to the special case of properly colored paths in edge-colored graphs. As finding a (simple) properly colored path between given endpoints in an edge-colored graph is polynomial-time solvable, we focus on the problem of finding a Hamiltonian cycle. We introduce a novel algebraic trick that shows that in edge-colored graphs, finding a properly colored Hamiltonian cycle is fixed-parameter tractable when parameterized by the treewidth of the graph. More specifically, the problem can be solved in time $2^{\mathcal{O}(k)} \cdot (|V(G)| + |V(\mathcal{T})| + \ell)$ where k is the treewidth, $\mathcal{T}$ is the tree of the decomposition and ℓ is the number of different colors the edges can have. The crucial property of the result is that ℓ, the number of colors, is *not* required to be bounded in the parameter and does not appear in the exponential part of the running-time bound (an exponential dependency on both k and ℓ is not hard to achieve).

After discussing graph-width notions, in Sect. 5, we move to the DISJOINT PATHS problem. In this problem, we are given a directed graph and a sequence $(s_1, t_1), (s_2, t_2), \ldots, (s_r, t_r)$ of terminal pairs; the goal is to find compatible paths $P_1, P_2, \ldots, P_r$ such that P_i starts in s_i and ends in t_i and the paths P_i are pairwise edge- or vertex-disjoint. In the undirected graph setting, the fixed-parameter tractability (parameterized by the number of requests r) of this problem is a milestone result in the Graph Minors series of Robertson and Seymour [48].

Observe that beyond undirected graphs the problem quickly becomes hard. As discussed, the setting of properly colored paths in edge-colored graphs generalizes directed graphs, and the DISJOINT PATHS problem for $r = 2$ is NP-hard in directed graphs [23]. Furthermore, in general graphs with transitions the case $r = 1$ is NP-hard. Hence, we focus on the specific case where the path P_i is required to be a shortest s_i-t_i path, even in the unrestricted graph. In directed graphs, a tractability result for this problem has been obtained by Bérczi and Kobayashi [9] for $r = 2$. This problem is currently a very active topic and new algorithms have been found very recently for several variants in the case $r = 2$. Polynomial algorithms have been developed by Gottschau et al. [26] and by Kobayashi and Sako [37] for undirected graphs with non-negative weighted edges and by Bang-Jensen et al. [3] in the directed unweighted case where paths do not have to be shortest but have bounded lengths. At the point of writing, the complexity of the problem was still open for $r \geq 3$. In the meantime, it was shown that finding r disjoint shortest paths in undirected, unweighted graphs is indeed polynomial-time solvable for each fixed r [8, 42].

In the light of the status described above, in this work we focus on the case $r = 2$ in directed forbidden-transition graphs. Extending the results of Bérczi and Kobayashi [9], we show that the problem remains polynomial-time solvable both in the edge- and vertex-disjoint case. The arguments are presented in Sect. 5.

2 Preliminaries

For each $n \in \mathbb{N}$ we use $[n]$ to denote $\{1, 2, \ldots, n\}$. Unless stated otherwise, all graphs are undirected, without self-loops and parallel edges.

2.1 Graphs

Let G be an undirected graph. By $V(G)$ and $E(G)$ we denote the vertex and edge set of G, respectively. For each $v \in V(G)$ we denote by $E_G(v)$ the set of edges in G that are incident with v in G. We omit the subscript G if it is clear from the context. A *walk* in G is a sequence $(v_1, e_1, v_2, e_2, \ldots, e_\ell, v_{\ell+1})$ where v_is are vertices of G, e_is are edges of G, and for every $1 \leq i \leq \ell$, the vertices v_i and v_{i+1} are the two endpoints of the edge e_i. A walk is *closed* if its first vertex is also its last vertex. The *length* of a walk W equals ℓ, the number of edges in W. A *path* is a walk in which no vertex occurs twice, a *cycle* is a closed walk in which no vertex occurs twice except the first and last vertex. Usually we will denote paths and cycles simply by their sequence of

vertices. By $\text{dist}_G(s,t)$ we mean the length of a simple s-t path in G (ignoring any transitions).

For a graph G, a *tree decomposition* of G is a pair $(\mathcal{T},\beta)$ where $\mathcal{T}$ is a tree and $\beta : V(\mathcal{T}) \to 2^{V(G)}$ such that the following holds: (i) for every $v \in V(G)$, the set $\{t \in V(\mathcal{T}) \mid v \in \beta(t)\}$ induces a nonempty connected subtree of $\mathcal{T}$, and (ii) for every $uv \in E(G)$, there exists $t \in V(\mathcal{T})$ with $u, v \in \beta(t)$. That is, the function β assigns to every node $t \in V(\mathcal{T})$ a subset $\beta(t) \subseteq V(G)$, often called a *bag*. It is often convenient to root $\mathcal{T}$ at an arbitrary vertex. The width of a tree decomposition $(\mathcal{T},\beta)$ equals $\max_{t\in V(\mathcal{T})} |\beta(t)| - 1$, and the treewidth of a graph is the minimum possible width of its tree decomposition.

2.2 Parameterized Complexity

A *parameterized problem* is a set of instances of the form (x,k), where $x \in \Sigma^*$ for a finite alphabet set Σ, and $k \in \mathbb{N}$ is the *parameter*. A parameterized problem Q is *fixed-parameter tractable* and in the class FPT, if there exists an algorithm that on input (x,k) decides if (x,k) is a yes-instance of Q in time $f(k)n^{\mathcal{O}(1)}$, where f is a computable function independent of $n = |x|$; an algorithm with this running time is called *fixed-parameter algorithm*. A *parameterized reduction* from a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ with parameter k to a parameterized problem $L' \subseteq \Sigma^* \times \mathbb{N}$ with parameter k' is a $g(k) \cdot |I|^{\mathcal{O}(1)}$-time computable function $f : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N} : (I,k) \to (I',k')$ such that $k' \leq h(k)$ for some computable function h and $(I,k) \in L \Leftrightarrow (I',k') \in L'$. A hierarchy of fixed-parameter intractability, *the W-hierarchy* $\bigcup_{t\geq 0} \text{W}[t]$, was introduced based on the notion of parameterized reduction, in which the 0th level W[0] is the class FPT. It is commonly believed that W[1] $\neq$ FPT. A parameterized problem Q is in the parameterized complexity class XP, if there exists an algorithm that on input (x,k) decides if (x,k) is a yes-instance of Q in time $f(k) \cdot n^{f(k)}$, where f is a computable function independent of $n = |x|$. For more discussion on parameterized complexity, we refer to the literature [15, 17, 22, 46].

3 Detours

As Szeider [49] proved, it is NP-hard to determine whether a given forbidden-transition graph (G,T) contains a compatible s-t path for two given vertices s and t. This of course implies that it is NP-hard to check whether there is a compatible s-t path of at most some given length. In contrast, it is polynomial-time solvable to decide whether there is a compatible s-t path which has length at most $\text{dist}_G(s,t)$. This can for example be seen by using the following strategy. Construct the line graph H of G. (That is, H has vertex set $E(G)$ and two vertices in H are adjacent if the corresponding edges in G share an endpoint.) For each vertex $v \in V(G)$ and each pair e, f of edges incident to v such that e and f are not compatible, remove the edge ef from H. Introduce two new vertices s' and t' into H and make them adjacent to every vertex corresponding to an edge incident in G with s or t, respectively. Finally, check whether H contains an (ordinary) s'-t' path of length at most $\text{dist}_G(s,t)+1$. However, we note that the above

method does not combine well with graph width parameters considered in this work (recall Fig. 1); if G is a star on $n+1$ vertices, both, its vertex cover and treecut-width are equal to one, while line graph H of G contains a clique of size n.

In this section we improve on the above observation by showing that checking for compatible s-t paths which are marginally longer than $\mathrm{dist}_G(s,t)$ can also be done efficiently. That is we are going to show the fixed-parameter tractability of the following problem.

ComDetour
Parameter: $k \in \mathbb{N}$

Input: An instance (G, T, s, t, k) where (G, T) is a forbidden-transition graph, $s, t \in V(G)$, and $k \in \mathbb{N}$.

Question: Does there exist a T-compatible s-t path in G of length at most $\mathrm{dist}_G(s,t)+k$?

For notational convenience, we slightly generalize the notion of an x-y path as follows. For a given graph G and $x, y \in V(G) \cup E(G)$, we say that a path $(v_1, v_2, \ldots, v_\ell)$ in G is an x-y path, if (i) $x = v_1 \in V(G)$ or $x = v_1v_2 \in E(G)$ and (ii) $y = v_\ell \in V(G)$ or $y = v_{\ell-1}v_\ell \in E(G)$.

We first show fixed-parameter tractability of the ComPath problem, which will be later used as a black box in our algorithm for ComDetour. The algorithm for ComPath uses a standard color-coding approach (see [15]), slightly modified to track the transitions.

ComPath
Parameter: $k \in \mathbb{N}$

Input: An instance (G, T, x, y, k) where (G, T) is a forbidden-transition graph, $x, y \in V(G) \cup E(G)$, and $k \in \mathbb{N}$.

Task: Decide whether there exists a T-compatible x-y path in G of length at most k. If so, return the length of a shortest such path.

The algorithm can be presented using random colorings but this would complicate the analysis later. We instead use the notion of perfect hash families. Let $n, k \in \mathbb{N}$ and let U be a set of size n. A *k-perfect hash family* of U is a family $\mathcal{F}$ of functions from U to $[k]$ such that for each subset $S \subseteq U$ of size at most k there exists a function $f \in \mathcal{F}$ such that $f(S) = [k]$ (that is, f is injective on S). Naor, Schulman, and Srinivasan [45] showed that a k-perfect hash family of size $e^k k^{\mathcal{O}(\log k)} \log n$ can be computed in $e^k k^{\mathcal{O}(\log k)} n \log n$ time; see also [15, Section 5.6.1].

Theorem 3.1 *There exists an algorithm solving* ComPath *on n-vertex graphs in time $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$.*

Proof Let (G, T, x, y, k) be the input instance. First, consider the case where $x, y \in V(G)$. The algorithm works as follows. We start by coloring the vertices in G using a family of perfect hash functions. Compute a $(k-1)$-perfect hash family $\mathcal{F}$ of $V(G)$ in $e^k k^{O(\log k)} n \log n$ time. Iterate over all elements $f \in \mathcal{F}$ and for each such element

f proceed as follows. Generate a coloring of $V(G)$ by starting with coloring x and y with two unique colors, say 1 and $k+1$ and then coloring the rest of vertices of G with colors $2, \ldots, k$, according to the values assigned by f. That is, put the color of a vertex $v \in V(G) \setminus \{x, y\}$ to be $f(v)+1$. Let $c : V(G) \to [k+1]$ be the resulting coloring and for each $i \in [k+1]$ denote by C_i the set $c^{-1}(i)$. Observe that it suffices to prove the following claim. □

Claim Let $x, y \in V(G)$. There exists an algorithm to test whether there exists a colorful T-compatible x-y path with at most $k+1$ vertices in time $2^k n^{\mathcal{O}(1)}$.

Proof (of Claim) We use the following dynamic programming approach that computes a table D: for a set S of at least two elements, such that $\{1\} \subseteq S \subseteq [k+1]$, and an edge $uv \in E(G)$ we define a boolean variable $D[S, u, v]$. We want it to be equal to TRUE if and only if there exists a colorful compatible x-uv path, whose vertices are colored with all colors from S.

For $S = \{1, i\}$ observe that $D[S, u, v] =$ TRUE if and only if $u = x$ and $v \in N(x) \cap C_i$. Thus, these entries of D can be computed in linear time. Next, for every S with $\{1\} \subseteq S \subseteq [k+1]$ and $|S| \geq 3$ and for every $e = uv \in E(G)$ we compute $D[S, u, v]$ as follows:

$$D[S,u,v] = \begin{cases} \bigvee \{D[S \setminus \{c(v)\}, w, u] : \\ \qquad \{wu, e\} \in T(u), wu \in E(G)\} & \text{if } c(v) \in S \setminus \{1\}, \\ \text{FALSE} & \text{otherwise.} \end{cases}$$

Observe that this is a correct way of computing the entries $D[S, u, v]$. As to the running time, the number of possible sets S is bounded by 2^k (as 1 is always included in S). Thus, $D[S, u, v]$ can be computed in $2^k n^{\mathcal{O}(1)}$ time. We say that $S \subseteq [k+1]$ is *good* if it contains 1 and $k+1$. Observe that a compatible x-y path of length at most k exists if and only if there exist $uv \in E(G)$ and a good set S, such that $D[S, u, v] =$ TRUE (note that in such a case $v = y$). Let $\mathcal{S}$ be the set of all triples (S, u, v) such that S is good and $D[S, u, v] =$ TRUE. If $\mathcal{S}$ is empty, then clearly there is no T-compatible x-y path of length at most k in G. Otherwise, we return the value of $|S| - 1$ for the smallest S such that $(S, u, v) \in \mathcal{S}$ for some $uv \in E(G)$.

As a result, the running time of the whole algorithm is $e^k k^{\mathcal{O}(\log k)} n \log n + e^k k^{\mathcal{O}(\log k)} \log n \cdot 2^k n^{\mathcal{O}(1)} = 2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$. Analogously, if we search for a colorful T-compatible x-y path when $\{x, y\} \cap E(G) \neq \emptyset$, we give unique colors to the second and/or last but one vertex of a potential path and slightly modify computation of the table D and we look for an optimal solution in $\mathcal{S}$ that respects the corresponding conditions. □

We are ready to prove the fixed-parameter tractability of ComDetour. Our algorithm for ComDetour is based on the work of Bezáková et al. [10] that shows that computing s-t paths of length exactly $\text{dist}(s, t) + k$ in ordinary graphs is fixed-parameter tractable with respect to k. The basic idea is that in such a path there are at most k segments in which no "progress" is made towards reaching t. These k segments can be determined locally by applying the algorithm for ComPath from above. The

remaining parts can be computed by ordinary dynamic programming as for computing shortest paths.

Theorem 3.2 *There exists an algorithm solving* ComDetour *in n-vertex graphs in time* $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$.

Proof Let (G, T, s, t, k) be the input instance of ComDetour and let $d = \text{dist}_G(s, t)$. We can clearly assume that $d > k$, as otherwise we can compute whether (G, T, s, t, k) is a yes-instance using the algorithm for ComPath with parameter $d + k \leq 2k$. For each $i \in [|V(G)|] \cup \{0\}$ we define the *i-th layer* $X_i := \{x \in V(G) : \text{dist}_G(s, x) = i\}$. Clearly $\{s\} = X_0$ and $t \in X_d$. Note that each compatible s-t-path of length at most $d + k$ must be contained in the graph induced by $X = \bigcup_{i\in\{0,1,\ldots,d+k\}} X_i$, therefore we can safely assume that $X = V(G)$. For two distinct layers X_i, X_j, we say that X_i is *higher* (resp. lower) than X_j if $i > j$ (resp $i < j$). We use the following notation: An edge $xy \in E(G)$ is called *inter-layer* if there exist $i, j \in [d + k] \cup \{0\}$ such that $i \neq j$, $x \in X_i$, and $y \in X_j$. If an edge is not inter-layer, we call it *within-layer*. For two vertices $p, q \in V(G)$ such that $\text{dist}(s, p) < \text{dist}(s, q)$ we denote by $G_{(p,q]}$ the subgraph of G induced by $\{p\} \cup \{x \in V(G) : \text{dist}(s, p) < \text{dist}(s, x) \leq \text{dist}(s, q)\}$. We also define $G_{(p,\infty)}$ to be the the subgraph of G induced by $\{p\} \cup \{x \in V(G) : \text{dist}(s, x) > \text{dist}(s, p)\}$.

The algorithm uses a dynamic-programming procedure that computes a table D. Table D is indexed by the inter-layer edges of G. For every inter-layer edge $xy \in E(G)$ such that y belongs to a higher layer than x, the entry $D[xy]$ contains the length ℓ of a shortest compatible xy-t path in $G_{(x,\infty)}$ if it exists and if $\text{dist}(s, x) + \ell \leq d + k$; otherwise $D[xy] = \infty$. Note that it suffices to compute the entries of the table D because we can then look up whether there is a solution in the entries corresponding to the edges incident with s (which are all inter-layer edges). We first compute the entries for edges with the highest layers and then for edges in successively lower layers. Intuitively, when filling D for a specific inter-layer edge uv, we can rely on the fact that each solution path using uv will contain an inter-layer edge wx in a higher layer and it will reach wx from uv in at most $2k$ steps. Thus, when filling the table for uv, we may refer to the correct entry for wx and compute the path between uv and wx using a call to the algorithm for ComPath.

At the beginning of the procedure we initialize the table, putting $D[xy] = \infty$ for every inter-layer edge $xy \in E(G)$. Next, we compute entries of D for the last $2k + 1$ layers: for every inter-layer edge $xy \in E(G)$ such that $\text{dist}_G(s, x) \geq d - k - 1$ and $\text{dist}_G(s, y) > d - k - 1$, solve the ComPath instance $(G_{(x,\infty)}, T, xy, t, 2k + 1)$. If for some xy the result is a path of length ℓ such that $\text{dist}_G(s, x) + \ell \leq d + k$, then we set $D[xy] = \ell$. Observe that this will fill $D[xy]$ with the correct value according to the definition of D.

Then, we inductively fill in earlier layers by carrying out the following computation steps:

(1) For every integer m from $d - k - 1$ down to 0 and for every pair of vertices x, u such that $\text{dist}_G(s, x) = m$ and $m < \text{dist}_G(s, u) \leq m + k + 1$ we do the following:

   (a) For every pair of edges $e, f \in E(G_{(x,u]})$ such that $e = xy$ (so e is an inter-layer edge) and $f = vu$ for some vertices $y, v \in V(G_{(x,u]})$, we do the following:

(i) We solve the ComPath instance $(G_{(x,u]}, T, e, f, 2k)$.
(ii) If the answer is negative, we continue with the next pair of candidates for e and f.
(iii) If the answer is positive, let r be the returned path length. Observe that $r \in [2k]$.
(iv) Let

$$p = \min\{D[g] \colon g \text{ is an inter-layer edge and } fg \in T(u)\}.$$

If $\mathrm{dist}(s, x)+r+p \leq d+k$ and $D[e] > r+p$, then we put $D[e] = r+p$.

For later reference, let $\nu = \min_{v \in N_G(s)} D[sv]$. We accept if and only if $\nu \leq \mathrm{dist}_G(s, t) + k$. Computing a single entry $D[xy]$ takes time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$, and since there are less than n^2 of them, $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ is the complexity of the whole algorithm.

We now show the correctness of the algorithm. Observe first, that each table entry $D[e]$ only receives a non-infinity value if there is a compatible e-t path of length at most $d + k$; this is ensured by the way the entries are filled in step (iv). Moreover, each non-infinity entry $D[e]$ contains the length of some compatible e-t path. Thus, the algorithm accepts only if there is a compatible s-t path of length at most $d + k$. In particular, if there exists no such path, then the answer is correct. Moreover, if there exists a compatible s-t path in G of length $\ell^\star \leq d + k$, then $\nu \geq \ell^\star$.

Now assume that there exists a compatible s-t path in G of length $\ell^\star$ such that $\ell^\star \leq d + k$. It remains to show that $\nu \leq \ell^\star$. We prove the stronger statement that for each inter-layer edge xy we have that $D[xy]$ is at most the length, ℓ, of a shortest compatible xy-t path in $G_{(x,\infty)}$ that satisfies $\mathrm{dist}_G(s, x) + \ell \leq d + k$ or ∞ if no such path exists. The proof is by induction on $d - m$ where m is the layer of x. By the above, the statement holds for $m \geq d - k$. Now assume that $m < d - k$. If there is no suitable compatible xy-t path, the statement clearly holds. Otherwise, let P be such a path. We claim that it suffices to show that on P there exist consecutive vertices v, u, z such that the following properties hold.

(P1) $m < \mathrm{dist}_G(s, u) \leq m + k + 1$.
(P2) $\mathrm{dist}_G(s, u) < \mathrm{dist}_G(s, z)$.
(P3) Let $P[u, t]$ be the subpath of P from u to t. Then $P[u, t]$ is contained in $G_{(u,\infty)}$.
(P4) There are at most $2k$ edges on P between (incl.) xy and vu.

Let $e = xy$, $f = vu$, and $g = uz$. If the above claim is true, then in Step (1) above we will guess x and u by (P1); in Step (a) we will guess e and f by definition of e; we will find an e-f path at most as long as the corresponding subpath of P in Step (i) by (P4); and we will consider $D[g]$ in the minimum in Step (iv) by (P2) and since P is compatible. Furthermore, $D[g]$ is at most the length, ℓ_u, of $P[u, t]$: By (P3), $P[u, t]$ is a path in $G_{(u,\infty)}$. To see that it satisfies the condition on its length let $\ell' = \ell - \ell_u$. Observe that $\mathrm{dist}_G(s, u) - \mathrm{dist}_G(s, x) \leq \ell'$ and thus we have $\mathrm{dist}_G(s, u) + \ell_u \leq \mathrm{dist}_G(s, x) + \ell' + \ell_u = \mathrm{dist}_G(s, x) + \ell \leq d + k$. Hence indeed, $P[u, t]$ satisfies $\mathrm{dist}_G(s, u) + \ell_u \leq d + k$, certifying that $D[g]$ is at most the length of $P[u, t]$. Thus $D[e]$ will receive a value that is at most the length of P in Step (iv), as required.

Before proving the claim, let us observe the following. Say that an inter-layer edge h is a *back* edge if P traverses the vertex in h that is in a larger layer before the other

vertex in h. Observe that the length of P is $d+a+2b$ where a is the number of within-layer edges in P and b the number of back edges in P. Thus, we have $a+b \leq k$. In particular, there are at most k layers in which P contains at least two vertices. We will use this fact below.

It remains to prove our claim above. Since there are at most k layers in which P contains at least two vertices, in the layers $m+1, m+2, \ldots, m+k+1$ there is at least one vertex, u, on P such that u is the only vertex of P in u's layer. We claim that u together with the vertex, v, that precedes u on P and the vertex, z, that succeeds u on P satisfy the properties in the claim. Clearly, (P1) is satisfied. Since u is the only vertex of P on u's layer, also (P2) is satisfied. For the same reason, (P3) is satisfied. Finally, suppose that (P4) does not hold, that is, there are more than $2k$ edges between xy and vu on P. Then the length, ℓ, of P is at least $2k+1+\mathrm{dist}_G(u,t)$. However, then $\mathrm{dist}_G(s,x)+\ell \geq \mathrm{dist}_G(s,x)+2k+1+\mathrm{dist}_G(u,t) > d+k$, a contradiction to the fact that $\mathrm{dist}_G(s,x)+\ell \leq d+k$. Thus, the claim holds, meaning that $v \leq \ell^\star$ and the algorithm is correct. □

4 Graph-Width Parameters

In this section we give our results pertaining to graph-width measures. As outlined in the introduction, finding a compatible s-t path of length at most k is fixed-parameter tractable with respect to k (see Theorem 3.1). Because the length of a simple path is upper-bounded by functions of the smallest size of a vertex cover and of the treedepth, tractability for these two parameters also follows. It is thus interesting to prove analogous tractability results for smaller and thus stronger parameters such as the treewidth of the input graph. However, in Sect. 4.1 we give a limit to this avenue: We prove that detecting compatible s-t paths and related problems are W[1]-hard with respect to the size of modulators to constant treewidth. In particular, this implies W[1]-hardness with respect to the pathwidth and the treewidth. Thus the natural dynamic-programming approaches that give fixed-parameter tractability in the ordinary graph setting likely do not work when taking transitions into account. It is interesting to note the contrast to the dynamic-programming approach for computing short detours (Theorem 3.2). The latter worked because the interface of a dynamic-programming enty consists of two single vertices and their transitions, whereas in the treewidth and pathwidth cases we would have to consider for each vertex in a separator all possible (unbounded number of) transitions.

On the positive side, if we focus on decompositional parameters that are based on edge cuts, rather than vertex separators, we can obtain tractability: In Sect. 4.2 we show that detecting compatible s-t paths is fixed-parameter tractable with respect to the treecut-width. Furthermore, the more restricted case of edge-colored graphs allows for efficient algorithms, too: In Sect. 4.3 we give a fixed-parameter algorithm for detecting properly colored Hamiltonian cycles in edge-colored graphs when parameterized by the treewidth.

4.1 Modulator to Linear Forest

Let G be an undirected graph. A *modulator to a linear forest* of G is a vertex subset $S \subseteq V(G)$ such that $G - S$ is a disjoint union of paths. The *distance k of G to a linear forest* is the minimum size, k, of a modulator to a linear forest. Note that the distance to a linear forest upper bounds the size of a minimum feedback-vertex set and the treewidth and hence W[1]-hardness for these two parameters is implied by W[1]-hardness for k. A *modulator to treewidth two* of G is a vertex subset $S \subseteq V(G)$ such that $G - S$ has treewidth at most two. The *distance of G to treewidth two* is the minimum size of a modulator to treewidth two. Analogously, the distance to treewidth two upper bounds the treewidth and hence W[1]-hardness for treewidth is implied by W[1]-hardness for the distance to treewidth two.

In this section, we first show that finding long paths or cycles is W[1]-hard with respect to the distance k to a linear forest. Moreover, assuming the Exponential Time Hypothesis (ETH), no $f(k) \cdot n^{o(k/\log k)}$-time algorithm can exist. Informally, the ETH states that 3- SAT on n-variable formulas cannot be solved in $2^{o(n)}$ time, see [32, 33]. We obtain the following.

Theorem 4.1 *Let (G, T) be forbidden-transition graph and s, t two vertices in G. Let ℓ be a positive integer and let k be the distance of G to a linear forest. For each of the following, it is W[1]-hard with respect to k to decide and, moreover, an $f(k) \cdot n^{o(k/\log k)}$-time decision algorithm contradicts the ETH:*

(i) whether G contains a compatible s-t path, and
(ii) whether G contains a compatible cycle.

Proof We first give a reduction to prove hardness of Item (i). We then modify the construction to obtain Item (ii).

Our reduction is from the Partitioned Subgraph Isomorphism (PSI) problem. Herein, we are given two graphs G and H, where $V(H) = [n_H]$ for some positive integer n_H, and a vertex coloring $\mathsf{col}\colon V(G) \to V(H)$ of the vertices of G with colors that one-to-one correspond to the vertices of H. Moreover, each vertex of H is incident with at least one edge and for each edge $\{u, v\} \in E(G)$ we have $\mathsf{col}(u) \neq \mathsf{col}(v)$. We want to decide whether H is isomorphic to a subgraph of G while respecting the colors, that is, whether there is an injective mapping $\phi\colon V(H) \to V(G)$ such that for all $u \in V(H)$ we have $\mathsf{col}(\phi(u)) = u$ and for all $\{u, v\} \in E(H)$ we have $\{\phi(u), \phi(v)\} \in E(G)$. In that case, we also say that ϕ is a *subgraph isomorphism* from H into G. In the following we let $m_H = |E(H)|$. Observe that $n_H \leq 2m_H$ since each vertex of H is incident with at least one edge. Since PSI contains Multicolored Clique [20] as a special case, PSI is W[1]-hard with respect to m_H. Moreover, Marx [43, Corollary 6.3] observed that an $f(m_H) \cdot n^{o(m_H/\log m_H)}$-time algorithm for PSI would contradict the ETH.

Intuitively, our construction works as follows (see also Fig. 2 for an illustration): We first build a path from s to a vertex t_1. This path is the concatenation of n_H subpaths $P^1, \ldots, P^{n_H}$ where each subpath is associated with a vertex of H. The subpath P^i contains a vertex for each edge of G incident to a vertex colored i. We then use an extra

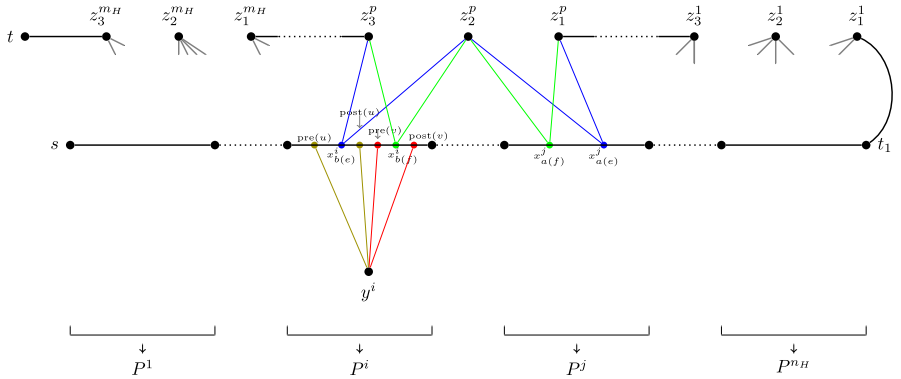

Fig. 2 An illustration of our construction. We use colors to denote edges that have to be used consecutively because of the set of permitted transitions. For example, the two dark yellow edges correspond to a vertex u of G in the vertex-selection gadget for vertex i of H and the four blue edges correspond to an edge e of G in the edge-selection gadget for the pth edge of H (Color figure online)

vertex and an appropriate transition system so that one can choose any vertex v of G with color i and connect the endpoints of P^i with a compatible path that skips exactly those vertices of P^i that denote an edge adjacent to v. This comes down to choosing $\phi(i) = v$. Finally, we connect t_1 to t by a sequence of gadgets each associated with an edge of H. Choosing a path through a gadget comes down to mapping an edge uv of H to an edge wx of G. Our transition system then requires the path in the gadget to visit the two vertices of P that denote the edge wx, which can only be done without repeating vertices if those vertices have been skipped between s and t_1. This means that the endpoints of wx have to be the vertices we chose as $\phi(u)$ and $\phi(v)$. By ensuring that there is an edge between $\phi(u)$ and $\phi(v)$, we prove that ϕ is a subgraph isomorphism. We now continue with the formal description.

Construction 4.1 Let (G, H, col) be an instance of PSI, where $V(H) = [n_H]$. For each $i \in [n_H]$ define $V_i = \{v \in V(G) \mid \mathsf{col}(v) = i\}$.

For each $i \in [n_H]$ define $E_i = \{e \in E(G) \mid \exists u \in V_i \colon u \in e\}$. We construct a forbidden-transition graph $(G^\star, T)$ as follows, see Fig. 2 for an illustration. We begin with $G^\star$ being empty. We will specify T by giving the permitted-transition sets $T(v)$ for the individual vertices $v \in V(G^\star)$. Below, we specify $T(v)$ only for a subset of $V(G^\star)$. For all the remaining vertices v, we put $T(v) = \binom{E(v)}{2}$ (recall that $E(v)$ is the set of edges in $G^\star$ that are incident with v). Introduce new vertices s, t, t_1 into $G^\star$. We construct the vertex-selection gadgets as follows.

Introduce a path P from s to t_1 into $G^\star$; we specify the number of vertices on P indirectly below. For each internal vertex $v \in V(P)$ put $T(v) = \{\{\{u, v\}, \{v, w\}\}\}$ where u and w are the neighbors of v in P. Additional edges and transitions for the vertices on P will be introduced below. Partition P into n_H disjoint paths $P^1, \ldots, P^{n_H}$; we specify the number of vertices in each of these paths in the next step.

For each $i \in [n_H]$, proceed as follows. Let $(e^i_a)_{a \in [r_i]}$ be an ordering of E_i such that, for each $v \in V_i$, the edges in $E(v)$ form a segment in (e^i_a) (observe that such an ordering exists since the endpoints of each edge in $E(G)$ have two different colors).

Set the number of vertices in P^i to $r_i + 4$. For each $a \in [r_i]$ denote the $a+2$-th vertex on P^i by x^i_a. We say that vertex x^i_a *corresponds* to the edge e^i_a of G. (We keep the first two and last two vertices of P^i unnamed.)

Next, introduce a vertex y^i and for each vertex $v \in V_i$ proceed as follows. Let $\mathsf{pre}(v)$ be the vertex in P^i that directly precedes on P^i the first vertex corresponding to an edge in $E_G(v)$. Similarly, let $\mathsf{post}(v)$ be the vertex in P^i that directly succeeds on P^i the last vertex corresponding to an edge in $E_G(v)$. For later, it is useful to observe that all vertices on P^i strictly between $\mathsf{pre}(v)$ and $\mathsf{post}(v)$ correspond to edges in $E_G(v)$. Now add the edges $\{\mathsf{pre}(v), y^i\}$ and $\{y^i, \mathsf{post}(v)\}$ to $G^\star$ and the transition $\{\{\mathsf{pre}(v), y^i\}, \{y^i, \mathsf{post}(v)\}\}$ to $T(y^i)$. Moreover, add the following transitions:

- $\{\{u, \mathsf{pre}(v)\}, \{\mathsf{pre}(v), y^i\}\}$ to $T(\mathsf{pre}(v))$ where u is the vertex on P^i preceding $\mathsf{pre}(v)$ (if any), and
- $\{\{y^i, \mathsf{post}(v)\}, \{\mathsf{post}(v), w\}\}$ to $T(\mathsf{post}(v))$, where w is the vertex on P^i succeeding $\mathsf{post}(v)$.

This finishes the construction of the vertex-selection gadgets, but further edges and transitions may be introduced later to the vertices of P.

We now construct the edge-verification gadgets. Let $(e_1, \ldots, e_{m_H})$ be an arbitrary ordering of the edges in $E(H)$. For each $p \in [m_H]$ proceed as follows. Introduce three vertices z^p_1, z^p_2, and z^p_3. Let $\{i, j\} = e_p$ where $i > j$. For each edge $e \in E_i \cap E_j$ of G proceed as follows. Let $a(e)$ be the index of e in the ordering (e^i_a) defined for vertex i when constructing the vertex-selection gadget. Similarly, let $b(e)$ be the index of e in the ordering (e^j_a) defined for j. Introduce the following edges into $G^\star$:

$$\{z^p_1, x^i_{a(e)}\}, \qquad \{x^i_{a(e)}, z^p_2\}, \qquad \{z^p_2, x^j_{b(e)}\}, \qquad \text{and} \qquad \{x^j_{b(e)}, z^p_3\}.$$

Furthermore, add the following transitions:

- $\{\{z^p_1, x^i_{a(e)}\}, \{x^i_{a(e)}, z^p_2\}\}$ to $T(x^i_{a(e)})$,
- $\{\{z^p_2, x^j_{b(e)}\}, \{x^j_{b(e)}, z^p_3\}\}$ to $T(x^j_{b(e)})$, and
- $\{\{x^i_{a(e)}, z^p_2\}, \{z^p_2, x^j_{b(e)}\}\}$ to $T(z^p_2)$.

To conclude the construction of the edge-verification gadgets, add the following edges: $\{t_1, z^1_1\}$; for each $p \in [m_H - 1]$ the edge $\{z^p_3, z^{p+1}_1\}$; and $\{z^{m_H}_3, t\}$. This concludes the construction of $G^\star$ and $T(G^\star)$ (recall that for vertices v for which we left $T(v)$ unspecified we put $T(v) = \binom{E(v)}{2}$).

Observe that Construction 4.1 can be carried out in polynomial time. We claim that the distance to linear forest of $G^\star$ is at most $n_H + 3m_H \leq 5m_H$. Let $Z = \{z^p_1, z^p_2, z^p_3 \mid p \in [m_H]\}$ and $Y = \{y^i \mid i \in [n_H]\}$. Note that the only vertices in $G^\star - (V(P) \cup \{t\})$ are in $Y \cup Z$. Moreover, no edges between two vertices on P have been introduced into $G^\star$. Thus, $Y \cup Z$ is a modulator to a linear forest and $G^\star$ has distance at most $5m_H$ to a linear forest. If Lemma 4.1 is correct, by the properties of PSI it thus follows that deciding whether a graph has a compatible s-t path is W[1]-hard with respect to the distance, k, to a linear forest, and that an $f(k)n^{o(k/\log k)}$-time decision algorithm contradicts the ETH. We next show the correctness of Lemma 4.1.

Correctness. We now show that $(G^\star, T)$ contains a compatible s-t path if and only if there is a subgraph isomorphism from H into G.

Suppose first that there is a subgraph isomorphism ϕ from H into G. Construct an s-t walk $P^\star$ by concatenating the following path segments (observe while reading the construction, that $P^\star$ is compatible):

1. The subpath on P from s to $\mathsf{pre}(\phi(1))$.
2. The three vertices $\mathsf{pre}(\phi(1))$, y^1, $\mathsf{post}(\phi(1))$.
3. For each $i = 2, 3, \ldots, n_H$ take:
   (a) The subpath on P from $\mathsf{post}(\phi(i-1))$ to $\mathsf{pre}(\phi(i))$.
   (b) The three vertices $\mathsf{pre}(\phi(i))$, y^i, $\mathsf{post}(\phi(i))$.
4. The subpath on P from $\mathsf{post}(\phi(n_H))$ to t_1.
5. For each $p = 1, 2, \ldots, m_H$, let e_p be the pth edge of H according to the ordering of $E(H)$ fixed in Lemma 4.1, let $e_p = \{i, j\}$, where $i > j$, let $e = \{\phi(i), \phi(j)\}$, let $a(e)$ be the index of e in the ordering (e_a^i) and $b(e)$ the index of e in the ordering (e_a^j). Take the vertices z_1^p, $x_{a(e)}^i$, z_2^p, $x_{b(e)}^j$, and z_3^p.
6. The edge $\{z_3^{m_H}, t\}$.

This concludes the construction of $P^\star$. Suppose, for a contradiction, that $P^\star$ is not a path, that is, there is a vertex v in $G^\star$ which is contained twice in $P^\star$. Since $V(G)$ is partitioned into $V(P)$, Y, Z, and $\{t\}$ and each vertex of Y and Z occurs only once in the definition of $P^\star$, we have $v \in V(P)$. Since each segment in the construction of $P^\star$ is a path, the two occurrences must be in different segments. Observe that all segments of $P^\star$ in steps 1 to 4 that are contained in $V(P)$ are pairwise disjoint subpaths of P. Furthermore, all vertices in $V(P)$ used in the segments constructed in step 5 are pairwise distinct. Thus, there is one occurrence of v in steps 1 to 4, and one in step 5. Moreover, v corresponds to some edge e of G. However, according to the steps 1 to 4, vertex v corresponds to some edge which is not incident to a vertex in $\phi(V(H))$ and, according to step 5, vertex v corresponds to some edge which is incident to a vertex in $\phi(V(H))$, a contradiction. Thus, indeed, $P^\star$ is a compatible s-t path, as required.

Now suppose that $(G^\star, T)$ contains a compatible s-t path $P^\star$. Obviously, $P^\star$ starts with a subsegment of P. By construction of the transitions on vertices on P, at each internal vertex of P, the path $P^\star$ may either continue on P or go to some vertex of Y. Moreover, whenever $P^\star$ traverses a vertex of Y, it immediately returns to P with the next vertex. Path $P^\star$ hence begins with a segment which starts at s, alternatingly contains a sequence of vertices on P and a vertex of Y, and ends at t_1. Let $Y' = Y \cap V(P^\star)$ (we show below that $Y' = Y$). Observe that, for each vertex $y^i \in Y'$, there exists $v \in V_i$ such that $P^\star$ contains the edges $\{\mathsf{pre}(v), y^i\}$ and $\{y^i, \mathsf{post}(v)\}$, by the transitions defined for y^i. Define a (partial) function $\phi\colon V(H) \to V(G)$ as follows. For each $i \in [n_H]$ such that $y_i \in Y'$ put $\phi(i) = v$, where v is as defined above. For later, put $P_1^\star$ to be the segment of $P^\star$ from s to t_1 and put $P_2^\star$ to be the segment of $P^\star$ from t_1 to t. Observe that $P_1^\star$ contains precisely all vertices of P except those that correspond to edges in G which are incident to the vertices of $\phi(Y')$.

To show that ϕ is total and that ϕ is a subgraph isomorphism from H into G, we now argue that $P_2^\star$ contains z_2^p for each $p \in [m_H]$. Since $P_2^\star$ is a path, it starts with the edge $\{t_1, z_1^1\}$. Moreover, by the edges and transitions of the vertices z_1^p, x_a^i, z_2^p, and

z_3^p ($p \in [m_H]$, $i \in [n_H]$, $a \in \mathbb{N}$), whenever $P_2^\star$ traverses a vertex z_1^p, $p \in [m_H]$, it next traverses some vertex x_a^i, then the vertex z_2^p, some vertex x_b^j, and the vertex z_3^p for some $i, j \in [n_H]$ where $i > j$. Moreover, after z_3^p, path $P_2^\star$ traverses either z_1^{p+1} (if $p < m_H$) or t (if $p = m_H$) because the only other vertices that $P_2^\star$ may traverse after z_3^p are vertices $x_{a'}^j$ and, by their transitions, $P_2^\star$ would then have to contain z_2^p a second time. Concluding, $P_2^\star$ contains z_2^p for each $p \in [m_H]$.

Let $p \in [m_H]$ and let e_p be the pth edge of H according to the ordering of $E(H)$ fixed in Lemma 4.1. Let $e_p = \{i, j\}$ with $i > j$. As argued above $P_2^\star$ contains z_2^p. Let x_a^i and x_b^j be the vertices that $P_2^\star$ traverses before and after z_2^p. By the transitions of z_2^p, the vertices x_a^i and x_b^j correspond to the same edge of G. Denote this edge by f_p. We now show that the edges f_p, $p \in [m_H]$, ensures that ϕ is total and a subgraph isomorphism.

First, to see that ϕ is total, recall that each vertex $i \in V(H)$ is incident with at least one edge. Say i is incident with edge e_p. Let x_a^i be the vertex that corresponds to an edge in G incident with a vertex of color i and that led to the definition of f_p, that is, $P_2^\star$ traverses x_a^i before or after z_p^2. Now recall that $P_1^\star$ contains all vertices of P except those that correspond to the edges incident with vertices in $\phi(Y')$. Since $P_1^\star$ and $P_2^\star$ are internally vertex-disjoint, $i \in Y'$. It thus follows that ϕ is total.

To see that ϕ is a subgraph isomorphism, take any edge $e_p \in E(H)$. Consider the edge f_p and the two vertices x_a^i and x_b^j that led to the definition of f_p, that is, x_a^i and x_b^j are traversed either before or after z_p^2. By the construction of the edges of z_p^2, we have $e_p = \{i, j\}$. We again use the property that $P_1^\star$ contains all vertices of P except those that correspond to the edges incident with vertices in $\phi(Y')$. Since x_a^i and x_b^j are not in $P_1^\star$, they correspond to an edge incident with both $\phi(i)$ and $\phi(j)$, that is, $f_p = \{\phi(i), \phi(j)\}$. Thus, indeed ϕ is a subgraph isomorphism, as required. This concludes the proof of Theorem 4.1 Item (i). The remaining parts are proved below.

Cycles. We now adapt Construction 4.1 to obtain Theorem 4.1 Item (ii). To this end, we simply add the edge $\{s, t\}$ to $G^\star$ (and update the permitted transitions of s and t to allow for combining $\{s, t\}$ with every other edge). Call the resulting graph $G_C^\star$. Observe that $G_C^\star - (Y \cup Z)$ is a path with vertex set $V(P) \cup \{t\}$, and hence $G_C^\star$ has distance to a linear forest at most $5m_H$.

We claim that there is a compatible s-t path in $G^\star$ if and only if there is a compatible cycle in $G_C^\star$. The forward direction is trivial. For the backward direction, let $C^\star$ be a compatible cycle in $G_C^\star$. We show that $C^\star$ contains $\{s, t\}$. For a contradiction, assume it does not. Thus, $C^\star$ is a cycle in $G^\star$. By the transitions of the vertices in P, cycle $C^\star$ does not contain an edge in P nor does it contain a vertex in Y. Let $G_1^\star = (V(G^\star) \setminus Y, E(G^\star) \setminus E(P))$ and observe that $C^\star$ is a cycle in $G_1^\star$. Observe that $V(P)$ is an independent set in $G_1^\star$. Thus each cycle (not necessarily compatible) can be written as $z_2^p, x_a^i, z_1^p, x_b^i, z_2^p$ or $z_2^p, x_a^i, z_3^p, x_b^i, z_2^p$ for the corresponding values of p, i, a, and b. However, by the transitions of z_2^p, none of these cycles is compatible, a contradiction. Thus, $C^\star$ contains $\{s, t\}$. Hence, removing $\{s, t\}$ from $C^\star$ gives an s-t path in $G^\star$, concluding the proof. □

We note that Theorem 4.1 immediately implies the following.

Corollary 4.1 *Let (G, T) be forbidden-transition graph and s, t two vertices in G. Let ℓ be a positive integer and let k be the distance of G to a linear forest. For each of the following, it is W[1]-hard with respect to k to decide and, moreover, an $f(k) \cdot n^{o(k/\log k)}$-time decision algorithm contradicts the ETH:*

(i) whether G contains a compatible s-t path of length at least ℓ (or at most ℓ), and
(ii) whether G contains a compatible cycle of length at least ℓ (or at most ℓ).

To put Corollary 4.1 in a wider context, note that, in contrast, deciding whether G contains an s-t path of length at least ℓ in ordinary graphs (without transitions) is fixed-parameter tractable with respect to treewidth (and, hence, also, parameterized by the distance to linear forest), see e.g., [15, Theorem 7.10].

We now adapt Construction 4.1 to prove that it is W[1]-hard with respect to the distance to treewidth two to check whether there is a compatible Hamiltonian cycle. This problem is, again, fixed-parameter tractable in ordinary graphs when parameterized by treewidth.

Theorem 4.2 *Let G be a graph and k' its distance to treewidth two. It is W[1]-hard with respect to k' to decide whether G contains a compatible Hamiltonian cycle and, moreover, an $f(k') \cdot n^{o(k'/\log k')}$-time decision algorithm contradicts the ETH.*

Proof To prove this theorem, we use Construction 4.1 and add a gadget that allows an s-t path in $G^\star$ to collect all so-far untraversed vertices, wherein we use transitions to not disturb the structure of $G^\star$. The basic observation that we use is that the path $P^\star$ we have constructed in the correctness proof for detecting s-t paths above contains all vertices of $G^\star$ except segments of the path P. The idea now is to add a path Q which runs "parallel" to P (like a skewed ladder) and which starts after t and ends in s. Using transitions we allow the solution in each vertex v of Q to either continue to the next vertex of Q or to traverse the vertex parallel to v on P and then immediately return to the next vertex after v on Q. This allows the solution to traverse all vertices it missed on the traversal from s to t. Since Q is parallel to P, removing $Y \cup Z$ will result in a graph of treewidth two.

The formal construction is as follows. Construct a forbidden-transition graph $(G_1^\star, T_1)$ from $(G^\star, T)$ by initially putting $(G_1^\star, T_1) = (G^\star, T)$. Let $n = |V(P)| - 2$. Add a path Q consisting of $n + 1$ vertices to $G_1^\star$ and identify the first and last vertex of Q with s and t, respectively. Let $v_1, v_2, \ldots, v_n$ be the internal vertices of P and $t = u_1, u_2, \ldots, u_{n+1} = s$ the vertices of Q. For each $i \in [n]$ proceed as follows. Add the edges $\{u_i, v_i\}$, and $\{v_i, u_{i+1}\}$. Then, update the transition system T_1 by adding the transitions $\{\{u_i, v_i\}, \{v_i, u_{i+1}\}\}$ to $T_1(v_i)$. This finishes the construction of $G_1^\star$ and its transition system (as before, for all vertices with unspecified transition systems we allow all transitions).

Let $\tilde{G_1^\star} = G_1^\star - (Y \cup Z \cup \{s, t\})$. We claim that $\tilde{G_1^\star}$ has treewidth two. Observe that this graph consists only of the vertices in P and Q except for s and t. Now observe that, by the definition of the edges between P and Q, the following bags give a path decomposition for $\tilde{G_1^\star}$ of width two. Note that we specify a bag containing $t = u_1$ for easier notation:

$$\{u_1, u_2, v_1\}, \{u_2, v_1, v_2\}, \ldots, \{u_i, u_{i+1}, v_i\}, \{u_{i+1}, v_i, v_{i+1}\}, \ldots,$$

$$\{u_{n-1}, u_n, v_{n-1}\}, \{u_n, v_{n-1}, v_n\}.$$

Thus $Y \cup Z \cup \{s, t\}$ is a modulator of $G_1^\star$ to treewidth two, meaning that $G_1^\star$ has distance to treewidth two at most $5m_H + 2$, as required.

We claim that $(G_1^\star, T_1)$ contains a compatible Hamiltonian cycle if and only if $(G^\star, T)$ contains a compatible s-t path. Take a compatible s-t path $P^\star$ in $(G^\star, T)$. Observe that in the correctness proof for detecting compatible s-t paths we have argued that $P^\star$ contains all vertices in Y and Z. In other words, the only vertices of $G_1^\star$ that $P^\star$ does not contain are in P and Q. To obtain a compatible Hamiltonian cycle in $(G_1^\star, T_1)$ from $P^\star$, simply extend the path after arriving at t by following Q and visiting vertices in $V(G_1^\star) \setminus Q$ as needed in order to visit all vertices of $G_1^\star$. In the other direction, by the updated transitions, a compatible Hamiltonian cycle in $G_1^\star$ decomposes into two s-t paths, one path containing vertices of Q (and a subset of vertices in P), and another path contained in $G^\star$, as claimed. This concludes the proof of W[1]-hardness of detecting compatible Hamiltonian cycles with respect to the distance to treewidth two and thus concludes the proof of Theorem 4.2. □

4.2 Treecut-Width

In this section we are going to prove that the ComPath problem is fixed-parameter tractable with respect to treecut-width of G; the treecut-width is defined below. In the ComPath problem we get a forbidden-transition graph (G, T_G) and two vertices $s, t \in V(G)$, and we want to decide whether there exists a compatible s-t path in G. The notion of treecut-width of a graph was introduced by Wollan [51]. In this section, to keep the notation succinct, we will sometimes refer to a forbidden-transition graph (G, T_G) simply as G and say that G *has* transition system T_G.

Basic definitions and previous results. We now define treecut-width and the associated treecut decompositions, and we recap the relevant previous results on computing them. Consider a graph G (without transitions) and let $v \in V(G)$ be a vertex of degree at most two. To *suppress* a vertex $v \in V(G)$ means (i) to add an edge between v's two neighbors (possibly creating a parallel edge) and (ii) to delete v. For a partition $A \cup B$ of the vertex set of G such that both A and B are non-empty, we denote by $E(A, B)$ the cut-set between A and B, i.e., the set $\{uv \in E(G) : u \in A, v \in B\}$. Recall that for a vertex v of G by $E_G(v)$ we denote the set of edges incident with v (we omit the subscript if it is clear from the context). We define *shrinking* a (not necessarily connected) set $Q \subseteq V(G)$ into q as an operation which replaces Q in G by a single new vertex q, and adds an edge qv for every edge $uv \in E(G)$ such that $u \in Q$ and $v \notin Q$. Note that this may create parallel edges.

A *treecut decomposition* of a graph G is a pair $(\mathcal{T}, \mathcal{X})$ such that $\mathcal{T}$ is a rooted tree and $\mathcal{X} = \{X_t\}_{t \in V(\mathcal{T})}$ is a partition of vertices of G in which we allow sets X_t to be empty. For a node t, let $\mathcal{T}_t$ be the subtree of $\mathcal{T}$ rooted in t. Let $Y_t := \bigcup_{t' \in V(\mathcal{T}_t)} X_{t'}$ and $Z_t := V(G) \setminus Y_t$. For a non-root node t, let $E_t := E(Y_t, Z_t)$, and for a root r, we set $E_r := \emptyset$. We denote the vertices from Y_t by $y_1, y_2, y_3, \ldots$, and the vertices from Z_t by $z_1, z_2, z_3, \ldots$.

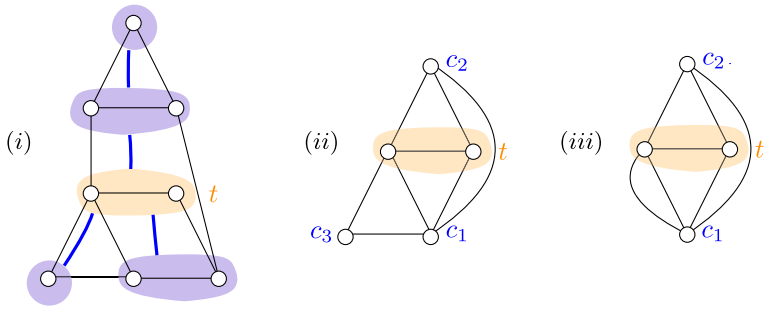

Fig. 3 **i** A graph G and its tree-cut decomposition $(\mathcal{T}, \mathcal{X})$ depicted in blue, with a distinguished node t. The left child of t is thin, while the right child is bold. **ii** The torso H_t of $(\mathcal{T}, \mathcal{X})$ at t, and **iii** the 3-center of H_t (Color figure online)

The *torso* of $(\mathcal{T}, \mathcal{X})$ at a node t is a graph H_t, constructed as follows. If $\mathcal{T}$ consists of a single node t, then the torso at t is G. Otherwise, let $C_1, \ldots, C_\ell$ be the sets of the vertices of G corresponding to the connected components of $\mathcal{T} \setminus \{t\}$. We construct the torso by shrinking C_i into c_i for each $i \in [\ell]$. Note that a torso may have parallel edges. The *3-center* $\widetilde{H}_t$ of a torso H_t is the graph obtained from H_t by suppressing vertices of degree at most two which belong to the set $V(H_t) \setminus X_t$. The *width* of a treecut decomposition $(\mathcal{T}, \mathcal{X})$ is $\max\{|E_t|, |V(\widetilde{H}_t)| : t \in V(\mathcal{T})\}$. The *treecut-width* of G is the minimum width of a treecut decomposition of G (see Fig. 3).

Theorem 4.3 *(Kim et al. [36]) There exists an algorithm that, given a graph G and $k \in \mathbb{N}$, either outputs a treecut decomposition of G of width at most $2k$ or correctly reports that G has treecut width larger than k in time $2^{\mathcal{O}(k^2 \log k)} \cdot |V(G)|^2$.*

A non-root node t of a treecut decomposition $(\mathcal{T}, \mathcal{X})$ is *thin* if $|E_t| \leq 2$ and it is *bold* otherwise. Denote by A_t and B_t, respectively, the set of all bold and thin children of t. A treecut decomposition of G is *nice* if for every thin node $t \in V(\mathcal{T})$ we have that $N(Y_t) \cap (\bigcup\{Y_b : b \text{is a sibling of} t \text{ in } \mathcal{T}\}) = \emptyset$.

Theorem 4.4 *(Ganian et al. [24]) There exists an algorithm working in time $\mathcal{O}(|V(G)|^3)$ which transforms any rooted treecut decomposition $(\mathcal{T}, \mathcal{X})$ of G into a nice treecut decomposition of the same graph, without increasing its width or number of nodes.*

The following property of nice decompositions is extremely useful in designing the dynamic programming algorithms.

Theorem 4.5 *(Ganian et al. [24]) Let t be a node of a nice treecut decomposition of width k. Then $|A_t| \leq 2k + 1$.*

Auxiliary problems and algorithms. To show that ComPath is fixed-parameter tractable with respect to the treecut-width we provide a dynamic-programming algorithm on the treecut decomposition. In the individual steps of the dynamic program we will need to solve the more general problem of finding compatible vertex-disjoint paths

between given pairs of vertices. We now introduce this problem and a restricted variant that occurs in a special case of the dynamic program, and we provide a fixed-parameter algorithm for the more restricted variant.

Let G be a graph. We say that W is *a set of terminal pairs* of G, if it consists of pairwise-disjoint two-element subsets of $V(G)$. If W is clear from the context, we also simply call the elements of W *terminal pairs* and each of them consists of *terminals*. Recall that by T_G we denote the transition system of G (again, we omit the subscript if it is clear from the context).

COMPATIBLE VERTEX- DISJOINT PATHS (COMVDP)
Parameter: The treecut-width, k, of G.
Input: An instance (G, T_G, W) where (G, T_G) is a forbidden transition graph and W is a set of terminal pairs of G.
Question: Are there pairwise vertex-disjoint T_G-compatible paths in G connecting each pair in W?

Below we will also say that a set of pairwise disjoint paths as above is a *solution* to the instance (G, T_G, W). Clearly, an instance (G, T_G, s, t) of COMPATH is equivalent to an instance $(G, T_G, \{\{s, t\}\})$ of COMVDP.

Our dynamic-programming approach is inspired by the XP-algorithm for finding edge-disjoint paths between given pairs of terminals, described by Ganian and Ordyniak [25]. However, we need modified dynamic-programming records and a more careful analysis to be able to ensure that subsolutions respect the transitions.

We first show fixed-parameter tractability of a simpler variant of COMVDP, called SCOMVDP, with additional assumptions on the structure of the input: Essentially, the vertex set is partitioned into a small set of arbitrary structure and a possibly large set of maximum degree two. A fixed-parameter algorithm for SCOMVDP will later be used when solving the general case.

SCOMVDP
Parameter: $|A| \in \mathbb{N}$
Input: An instance (G, T_G, W) of COMVDP and a partition of $V(G)$ into two sets A, B such that B consists of vertices of degree at most 2.
Question: Are there pairwise vertex-disjoint T_G-compatible paths in G connecting each pair in W?

Observe that each instance of SCOMVDP has treecut-width at most $|A|$. Indeed, we may construct a treecut decomposition of G as follows: All vertices of A are contained in a bag X_r and each vertex $b \in B$ forms a separate bag X_b. We define $\mathcal{T}$ to be a star, with r as its center and root. It is straightforward to verify that $(\mathcal{T}, \{X_t\}_{t \in \{r\} \cup B})$ is a treecut decomposition of G of width $|A|$.

We now generalize the notion of suppressing a vertex to forbidden-transition graphs. Let (G, T) be a forbidden-transition graph and v a vertex of degree at most 2 in G. If v is of degree 0 or 1, or $T(v)$ does not contain any transition, we delete v. Otherwise, $T(v)$ consists of a single transition, say $\{uv, vw\}$.

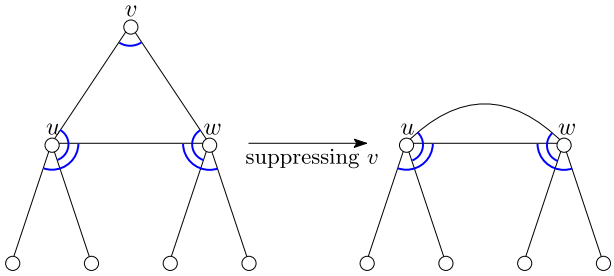

Fig. 4 An operation of suppressing a vertex v. The transition systems are represented by blue arcs near common vertices of the edges (Color figure online)

- If $uw \notin E(G)$, then we proceed as follows. We add uw to $E(G)$. In each transition of $T(u)$ we replace uv with uw and in each transition of $T(w)$ we replace vw with uw. That is, if there is a transition t in $T(u)$ (resp. in $T(w)$) with $t = \{uv, ux\}$ (resp. with $t = \{vw, wx\}$) for a vertex $x \in V(G) \setminus \{u, v, w\}$, then we put $T(u) := T(u) \setminus \{\{uv, ux\}\} \cup \{\{uw, ux\}\}$ (resp. $T(w) := T(w) \setminus \{\{vw, wx\}\} \cup \{\{uw, wx\}\}$).
- If $uw \in E(G)$, then we add an edge uw_v, parallel to uw. Then, for each vertex $x \in V(G) \setminus \{u, v, w\}$, if $\{ux, uv\} \in T(u)$ then we add $\{ux, uw_v\}$ to $T(u)$ and if $\{wx, vw\} \in T(w)$ then we add $\{wx, uw_v\}$ to $T(w)$.

Regardless of whether $uw \in E(G)$, we remove v from G (and remove the corresponding transitions).

Note that in ordinary graphs (without transitions), deleting a parallel edge preserves the existence of a solution that consists of pairwise vertex-disjoint edges. However, in our case we need parallel edges to handle transitions, see Fig. 4. Using them, the transitions in a forbidden-transition graph (G', T') obtained from (G, T) by suppressing v such that $N_G(v) = \{u, w\}$ are defined in such a way that we are allowed to use the original edge uw of G in a compatible path S in (G', T') if and only if S is a compatible path also in (G, T) or a path obtained from S by replacing the edge uw by two consecutive edges uv, vw is compatible in (G, T). This implies that suppressing a non-terminal vertex v is a safe reduction rule, that is, it preserves the existence of a solution to ComVDP or SComVDP.

We are now ready to solve SComVDP.

Lemma 4.2 *There exists an algorithm solving* SComVDP *in time* $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$, *where* $k = |A|$ *and* $n = |V(G)|$.

Proof Let $J = (G, T, W)$ be an instance of SComVDP. We start with simple sanity checks. First, observe that if $|W| > n$ then J is clearly a no-instance as we cannot find more than n vertex-disjoint paths in G. Similarly, if there exists a vertex which belongs to more than one pair in W, then J must be a no-instance. Performing the sanity checks takes $\mathcal{O}(n^2)$ time.

Consider a vertex $v \in B$. If v does not belong to any pair in W and is not adjacent to any vertex by a parallel edge, then we suppress it. Recall that this preserves the solution. If v does not belong to any pair in W but is adjacent to a vertex by a parallel edge, we remove it, as it cannot be used by any path in a solution.

Therefore, we can assume that for each vertex $v \in B$ there exists a unique vertex $v' \in V(G)$ with $v' \neq v$ such that $\{v, v'\} \in W$. For every $v \in B$ such that $N(v) \subseteq B$, we check whether $v' \in N(v)$. If yes, we can safely remove v and v' from G and $\{v, v'\}$ from W. Otherwise, we report that J is a no-instance – since all vertices in B are terminal vertices, there is no way to connect v and v'. After this step, each vertex in B must have a neighbor in A. Moreover, observe that there is no $\{v, v'\} \in W$ such that $v, v' \in B$ and $v' \in N(v)$, so each path in a potential solution must intersect A.

Next, if $|B| > 2|A|$ we report that J is a no-instance. Indeed, in order for J to be a yes-instance, since each vertex in B is a terminal, and each path in the solution containing a vertex of B must also contain a vertex of A, for each two vertices in B there must exist a vertex in A. Thus, $|V(G)| \leq 3k$, and an algorithm guessing the partition of the vertex set into the desired paths solves the instance in $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time (as the number of edges in G is bounded by n^2). The statement now follows. □

We emphasize that solving SComVDP will be the only moment where non-simple graphs (that is, graphs with parallel edges) appear in our algorithm. Since it will be used as a last step, to solve a reduced instance of the original problem, in the remaining part we describe the procedure only for simple graphs.

Before we proceed with the algorithm, we introduce a notion which will be useful when handling transitions. Let G' be an induced subgraph of G. We say that a family $\mathcal{C} := \{C_1, \ldots, C_\ell\}$ consisting of disjoint subsets of $E\big(V(G'), V(G) \setminus V(G')\big)$ is *terminable* (with respect to G') if for every $i \in [\ell]$ set C_i is either (i) a singleton or (ii) contains exactly two edges uu' and vv' such that $u', v' \in V(G')$ and $u' \neq v'$. We omit reference to the graph G that contains G' if it is clear from the context. We now define the operation of terminating a terminable set $\mathcal{C}$. Intuitively, this operation returns a modified G' in which all edges in the subsets of $\mathcal{C}$ are added and, if two edges were from the same subset of $\mathcal{C}$, then their endpoints outside of G' are merged into one vertex and the transition over this vertex is made to be allowed (see Fig. 5).

Definition 4.1 Let G be a simple graph with transition system T_G. Consider an induced subgraph G' of G and let $\mathcal{C} := \{C_1, \ldots, C_\ell\}$ be terminable with respect to G'. We define the operation of *terminating* $\mathcal{C}$ *in* G' which results in a graph $G'_{\mathcal{C}}$ with transition system $T_{G'_{\mathcal{C}}}$ as follows. The resulting graph $G'_{\mathcal{C}}$ has vertex set $V(G') \cup V'$, where the elements of V' are $c(C_1), \ldots, c(C_\ell)$. The edge set of $G'_{\mathcal{C}}$ is defined as follows.

$$E(G'_{\mathcal{C}}) := E(G') \cup \{uc(C_i) : u \in V(G') \text{ and there exists an edge in } C_i \text{ adjacent to } u \text{ in } G\}.$$

For simplicity we will sometimes write c_i instead of $c(C_i)$, if it causes no confusion.

The transition system $T_{G'_{\mathcal{C}}}$ of $G'_{\mathcal{C}}$ is defined as follows.

- If $u \notin V'$, then for $w, z \in N(u)$ a transition $\{wu, uz\}$ belongs to $T_{G'_{\mathcal{C}}}(u)$ if and only if
  - $w, z \notin V'$ and $\{wu, uz\} \in T_G(u)$, or
  - $w \notin V'$, $z = c_i \in V'$ and $\{wu, e\} \in T_G(u)$, where e is the unique edge from C_i adjacent to u, or

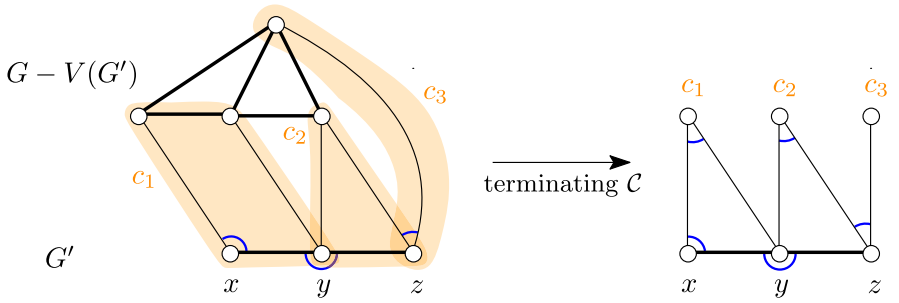

Fig. 5 An example of terminating a family $\mathcal{C}$ (orange) with respect to G' (Color figure online)

- $w = c_i, z = c_j \in V'$, for $i \neq j$, and $\{e, f\} \in T_G(u)$, where e and f are, respectively, the unique edges from C_i and C_j adjacent to u.

– If $u = c_i \in V'$, then $T_{G'_{\mathcal{C}}}(u)$ contains all unordered pairs of edges incident with u.

Note that since each C_i has at most 2 elements, all vertices from V' have degree at most 2.

Observe that after terminating some $\mathcal{C}$ in G', we always obtain a simple graph. Moreover, the following observation is straightforward.

Observation 4.3 *Let G' be an induced subgraph of G and let $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ be a terminable set with respect to G'. Let $P = (p_1, p_2, \ldots, p_m)$ be a compatible path in G such that at least one p_i belongs to $V(G')$. Denote by $e_1, e_2, \ldots, e_r$ the edges of $E(V(G'), V(G) \setminus V(G')) \cap P$ in the order in which they appear in P.*

1. *If $p_1, p_m \in V(G')$ and for each odd $j \in [r-1]$ we have $\{e_j, e_{j+1}\} \in \mathcal{C}$ then there exists a compatible p_1-p_m path in $G'_{\mathcal{C}}$.*
2. *If $p_1 \in V(G')$ (resp. $p_1 \notin V(G')$), and for some $i, i' \in [\ell]$ and even (resp. odd) $j \in [r-1]$ we have $C_i = \{e_j\}, C_{i'} = \{e_{j+1}\} \in \mathcal{C}$, then there exists a compatible c_i-$c_{i'}$ path in $G'_{\mathcal{C}}$,*
3. *If for some $i \in [\ell]$ we have $C_i = \{e_1\} \in \mathcal{C}$ (resp. $C_i = \{e_r\} \in \mathcal{C}$), then there exists a compatible p_1-c_i path (resp. c_i-p_m path) in $G'_{\mathcal{C}}$.*

Algorithm overview. The algorithm will compute the solution to ComVDP by dynamic programming over $(\mathcal{T}, \mathcal{X})$ in a leaf-to-root fashion. In a fixed node t of $\mathcal{T}$, we consider the possible partitions of E_t into four sets I (*internal*), F (*foreign*), L (*leaving*), and U (*unused*). Intuitively, the edges in sets I, F, and L will correspond to edges in the solution paths: paths with both terminals in Y_t use edges in I, paths with both terminals in Z_t use edges in F and paths with one terminal $y \in Y_t$ and another one $z \in Z_t$ use one edge of L (which is supposed to be the first edge on the solution path from y to z which belongs to E_t) and their other edges should belong to F. For each such partition we consider all possible *records*, that is, ways in which a potential solution could behave on E_t (see Definition 4.2).

If now t is a leaf, for every such record we check whether it is *valid*, i.e., whether we can extend the potential solution corresponding to it to a graph created from the

input graph G by terminating some family of subsets of E_t with respect to $G[X_t]$. This can be done using an algorithm that solves SComVDP. Otherwise, t has a child t'. If t' is thin, we use a reduction rule to handle it. If t' is bold, for each valid record of t', we perform a *simplification* – we terminate some family of subsets of $E_{t'}$ with respect to $G[Z_{t'}]$. Then, after simplifying every bold child of t, we can again check whether it is valid, by terminating some family of subsets of E_t and solving SComVDP.

Notions for dynamic programming. We now introduce the notions used in our dynamic-programming approach and give results that we will later need to prove the correctness. Let (G, T_G, W) be an instance of ComVDP and let $(\mathcal{T}, \mathcal{X})$ be a treecut decomposition of G of width k. For a set $X \subseteq V(G)$, by $W[X]$ we denote the subset of terminal pairs from W with both elements in X. For a node $t \in V(\mathcal{T})$ let $G_t = G[Y_t]$. An *unmatched terminal* for t is a vertex $x \in Y_t$ such that there exists a vertex $y \in Z_t$ satisfying $\{x, y\} \in W$. We let U_t be the set of unmatched terminals for t. We will use the fact that the number of unmatched terminals is bounded by the width of $(\mathcal{T}, \mathcal{X})$ in every yes-instance:

Observation 4.4 *If (G, T_G, W) is a yes-instance, then for each node t of $\mathcal{T}$ the number $|U_t|$ of unmatched terminals is at most k.*

Proof Consider a solution to the instance and observe that this solution witnesses that there is a flow between U_t and Z_t of value at least $|U_t|$. Since E_t is a Y_t-Z_t cut in G containing at most k edges and $U_t \subseteq Y_t$, we have $|U_t| \leq |E_t| \leq k$. □

Since Observation 4.4 is easily checkable in polynomial time, we will from now on assume that for each node t of $\mathcal{T}$ we have $|U_t| \leq k$.

As mentioned, we are going to describe a dynamic-programming procedure on the treecut decomposition $(\mathcal{T}, \mathcal{X})$. Below we introduce a basic notion which will be used to store the information about partial solutions.

Definition 4.2 A *record* R for $t \in V(\mathcal{T})$ is a tuple $(\sigma, \mathcal{I}, \mathcal{F}, \lambda)$ consisting of the following elements.

- Function σ is a partition of E_t into sets I (*internal*), F (*foreign*), L (*leaving*), and U (*unused*), such that for every $v \in V(G)$ we have $|E_G(v) \cap (I \cup F \cup L)| \leq 2$. Moreover, for each vertex $v \in V(G)$ with $|E(v) \cap (I \cup F \cup L)| = 2$ either (i) both elements of $E(v) \cap (I \cup F \cup L)$ belong to exactly one of I or F, or (ii) $v \in Z_t$ and one element of $E(v) \cap (I \cup F \cup L)$ belong to to F and another one to L.
- Set $\mathcal{I}$ is a perfect matching between the elements from I, such that if $\{y_i z_i, y_j z_j\} \in \mathcal{I}$ then $y_i \neq y_j$ (recall that y_i, y_j denote vertices in Y_t and z_i, z_j vertices in Z_t).
- Set $\mathcal{F}$ is a perfect matching between the elements from F, such that if $\{y_i z_i, y_j z_j\} \in \mathcal{F}$ then $z_i \neq z_j$.
- Finally, λ is a bijection between U_t and L.

Observe that the conditions on σ together with the conditions on the matchings $\mathcal{I}$ and $\mathcal{F}$ ensure that $\mathcal{I} \cup \{\{e\} \colon e \in F \cup L\}$ is terminable with respect to the subgraph $G[Y_t]$ of G and that $\mathcal{F} \cup \{\{e\} \colon e \in I \cup L\}$ is terminable with respect to the subgraph $G[Z_t]$ of G.

Let $R(t)$ denote the set of all possible records for t. Observe that $|R(t)| \leq 4^k \cdot (k!)^3 = k^{O(k)}$, as there are at most 4^k possibilities for σ, there are at most $k!$ possibilities each

for the matchings $\mathcal{I}$ and $\mathcal{F}$, and at most $k!$ possibilities for λ because there are at most k unmatched terminals by Observation 4.4.

Intuitively, the matchings should capture, in case of edges in I, which edge is used by path to leave Y_t and then to come back, and in case of edges in F, by which edge we enter Y_t and which one is then used to leave. Finally, λ says by which edge we leave Y_t for the first time. Below we introduce a notion which will help to formalize this intuition.

Definition 4.3 For an instance (G, T, W), a node t of a treecut decomposition $(\mathcal{T}, \mathcal{X})$ of G and a record $R = (\sigma, \mathcal{I}, \mathcal{F}, \lambda) \in R(t)$, we construct a *corresponding instance* (G_R, T_R, W_R) of ComVDP as follows. Let $\mathcal{C} = \{C_1, \ldots, C_\ell\} := \mathcal{I} \cup \{\{e\} : e \in F \cup L\}$. Let the graph G_R, together with the transition system T_R, be obtained by terminating $\mathcal{C}$ in G_t. Denote by $V_R = \{c(C_1), \ldots, c(C_\ell)\}$ the set of vertices $V(G_R) \setminus V(G_t)$. The set W_R contains

(i) every element of $W[Y_t]$,
(ii) the pair $\{c_i, c_j\}$ for every $c_i, c_j \in V_R$ such that $C_i \cup C_j \in \mathcal{F}$, and
(iii) the pair $\{a, c_i\}$ for every $a \in U_t$ and $c_i \in V_R$ such that $C_i = \{\lambda(a)\}$.

Note that the set $\mathcal{C}$ was defined in such a way that the pairs added to W_R are disjoint.

A record R is *valid*, if its corresponding instance (G_R, T_R, W_R) is a yes-instance of ComVDP. A corresponding instance should capture how the potential solution we construct behaves on G_t.

Definition 4.4 Let $(\mathcal{T}, \mathcal{X})$ be a treecut decomposition of G, let $t \in V(\mathcal{T})$ and let $J = (G, T, W)$ be an instance of ComVDP. Assume that there exists a solution $S = \{P_1, P_2, \ldots, P_{|W|}\}$ to J. We say that solution S *corresponds* to a record $(\sigma, \mathcal{I}, \mathcal{F}, \lambda)$ for the node t, if the following conditions are satisfied for every a_i-b_i path $P_i \in S$ such that $E(P_i) \cap E_t \neq \emptyset$. Let $e^i_1, e^i_2, \ldots, e^i_{r_i}$ denote the elements of $P_i \cap E_t$ in the order of their appearance on P_i.

1. If $a_i, b_i \in Y_t$, then $\{e^i_j, e^i_{j+1}\} \in \mathcal{I}$ for each odd $j \in [r_i - 1]$.
2. If $a_i, b_i \in Z_t$, then $\{e^i_j, e^i_{j+1}\} \in \mathcal{F}$ for each odd $j \in [r_i - 1]$.
3. If $a_i \in Y_t, b_i \in Z_t$, then $\lambda(a_i) = e^i_1$ (note that in this case a_i is an unmatched terminal, that is, $a_i \in U_t$) and $\{e^i_j, e^i_{j+1}\} \in \mathcal{F}$ for each even $j \in [r_i - 1]$.
4. An edge $e \in E_t$ belongs to U if and only if $e \notin \bigcup_{i \in [|W|]} E(P_i)$.

The dynamic-programming procedure on the treecut decomposition $(\mathcal{T}, \mathcal{X})$ of G computes the set $D(t)$ of valid records for a node t of $\mathcal{T}$. In other words, in $D(t)$ we store the information about these behaviors of a potential solution on E_t, which can be extended to G_t. Note that (G, T, W) is a yes-instance of ComVDP if and only if $D(r) = \{(\emptyset, \emptyset, \emptyset, \emptyset)\}$.

The last operation which will be used in the algorithm is the simplification of an instance. Intuitively, for a record R, the simplified instance is a smaller instance of the ComVDP problem which is equivalent to the original instance J assuming that, if J is an yes-instance, then R corresponds to some solution for J. Note that the construction of the simplified instance can be seen as dual to the construction of the corresponding instance.

Definition 4.5 Let $(\mathcal{T}, \mathcal{X})$ be a treecut decomposition of G, let $t \in V(\mathcal{T})$ and let $J = (G, T, W)$ be an instance of ComVDP. The operation of *simplification* of the instance J in node t in accordance with record $R = (\sigma, \mathcal{I}, \mathcal{F}, \lambda) \in R(t)$ returns an instance (G_Q, T_Q, W_Q) as follows. Let $\mathcal{C} = \{C_1, \ldots, C_\ell\} := \mathcal{F} \cup \{\{e\} : e \in I \cup L\}$. Graph G_Q and its transition system T_Q are obtained by doing the termination of $\mathcal{C}$ with respect to $G[Z_t]$. Let $V_Q := \{c(C_1), \ldots, c(C_\ell)\}$. The set of terminal pairs W_Q contains

(i) every element of $W[Z_t]$,
(ii) the pair $\{c_i, c_j\}$ for every $c_i, c_j \in V_Q$ such that $C_i \cup C_j \in \mathcal{I}$, and
(iii) the pair $\{c_i, b\}$ for every $\{a, b\} \in W$ with $a \in U_t$, $b \in Z_t$, and $c_i \in V_Q$ such that $C_i = \{\lambda(a)\}$.

Observe that each vertex in V_Q has degree at most 2, and the degree of vertex in $V(G_Q) \setminus V_Q$ is at most its degree in G.

The following lemmata reveal how the introduced notions are related to each other.

Lemma 4.5 *Let $(\mathcal{T}, \mathcal{X})$ be a treecut decomposition of G and let $J = (G, T, W)$ be an instance of* ComVDP *which admits a solution S. Then for every node $t \in V(\mathcal{T})$ there exists a unique record $R \in R(t)$ such that S corresponds to R.*

On the other hand, if S corresponds to some record R, then R must be valid.

Proof It is clear that for a fixed S and t there exists an unique $R = (\sigma, \mathcal{I}, \mathcal{F}, \lambda) \in R(t)$ satisfying conditions 1.-4. in the Definition 4.4.

The fact that there exists a solution to the corresponding instance of R follows from the construction of the corresponding instance and Observation 4.3. □

Lemma 4.6 *Consider an instance $J = (G, T, W)$ of* ComVDP. *Let $(\mathcal{T}, \mathcal{X})$ be a treecut decomposition of G, let t be a fixed node of $\mathcal{T}$, and let $R = (\sigma, \mathcal{I}, \mathcal{F}, \lambda) \in R(t)$. Let J_Q be the result of the simplification of J in t in accordance with R.*

1. *If J_Q admits a solution and R is valid, then J admits a solution.*
2. *If S is a solution to J and S corresponds to R, then J_Q admits a solution.*

Proof Let $J_Q = (G_Q, T_Q, W_Q)$. Recall that $V(G_Q) = V(G[Z_t]) \cup V_Q$, $V_Q = \{c(C_1), c(C_2), \ldots, c(C_\ell)\}$ and $\mathcal{C} = \{C_1, C_2, \ldots, C_\ell\}$ is a set terminated with respect to $G[Z_t]$. Note that in both statements of the lemma, R must be valid – in the first one by assumption and in the second one it follows from Lemma 4.5. Let thus $J_R = (G_R, T_R, W_R)$ be a yes-instance corresponding to R, obtained by terminating $\mathcal{C}' = \{C'_1, C'_2, \ldots, C'_{\ell'}\}$. Let $V_R = \{c'(C'_1), c'(C'_2), \ldots, c'(C'_{\ell'})\}$ (we will also write c'_i instead of $c'(C'_i)$) and let S_R be a solution to J_R.

For the first statement, assume that S_Q is a solution to J_Q. For each $\{a, b\} \in W$ we construct a compatible a-b path P^* to include in a solution for J as follows. We iteratively construct a sequence P^* of elements of $V(G_R) \cup V(G_Q)$ with the property that, at each iteration, two consecutive vertices of P^* either form an edge from $E(G_R) \cup E(G_Q)$ or from E_t. We claim that, at the end of the procedure, P^* is a compatible a-b path in G.

We first consider the case in which $a, b \in Y_t$. Observe that there is an a-b path $P \in S_R$ in G_R. We start with P^* being the consecutive vertices of P. If there are no

vertices from the set V_R in P, then P^* is the desired a-b path in G. Otherwise, we proceed to Step 1.

Step 1 (Replacing vertices from V_R): Denote by $e_1, e_2, \ldots, e_m$ the edges from $E(Y_t, V_R) \cap E(P)$ in the order in which they appear on P. Since $a, b \in Y_t$ and by the construction of J_R we must have $\{e_1, e_2\}, \{e_3, e_4\}, \ldots, \{e_{m-1}, e_m\} \in \mathcal{I}$. This implies that for each $i \in \{1, 3, \ldots, m-1\}$ there exists a $c(\{e_i\})$-$c(\{e_{i+1}\})$ path in S_Q; call this path $P_{i,i+1}$. We replace each element of V_R adjacent to edges e_i and e_{i+1} in P^* by the interior vertices of $P_{i,i+1}$. Observe that the result respects the transitions of G by the definition of termination of a set. Observe that in this way we may have added some vertices from V_Q to the sequence P^*. If this has happened, we take care of them in Step 2.

Step 2 (Replacing vertices from V_Q): Let $e'_1, e'_2, \ldots, e'_{m'}$ be the edges from $E(Z_t, V_Q) \cap E(P_{i,i+1})$ in the order in which they appear on $P_{i,i+1}$. By the construction of J_Q we have $\{e'_1, e'_2\}, \ldots, \{e'_{m'-1}, e'_{m'}\} \in \mathcal{F}$. Moreover, for each $j \in \{1, 3, \ldots, m'-1\}$ there exists a $c'(\{e'_j\})$-$c'(\{e'_{j+1}\})$ path $P_{j,j+1}$ in S_R. We replace each element of V_Q adjacent to edges e'_j and e'_{j+1} in P^* by the interior vertices of the $c(\{e'_j\})$-$c(\{e'_{j+1}\})$ path $P'_{j,j+1}$ from S_R. Observe again that the result respects the transitions of G by the definition of termination of a set. Again, in Step 2 we can add to the sequence some vertices from V_R. If this happens, we go back to Step 1.

Since the paths in S_R and S_Q are pairwise disjoint, we never add a vertex to P^* twice. Moreover, $|V_R|, |V_Q| \leq k$, so after at most $2k$ iterations, we obtain an a-b path P^* which uses only vertices from $Y_t \cup Z_t$, as required.

The case in which $a, b \in Z_t$ is analogous; the only difference is that the initial path P^* is taken from S_Q, and therefore it can contain vertices from V_Q and, in that case, we start with Step 2. If $a \in Y_t$ and $b \in Z_t$, we start with the $c'(\{\lambda(a)\})$-b path P^* from S_Q and, before performing Step 1, we replace $c'(\{\lambda(a)\})$ in P^* by the vertices of the a-$c(\{\lambda(a)\})$ path in S_R.

It is straightforward to verify that we use every path from S_R and S_Q at most once when we construct paths for all terminal pairs $\{a, b\}$. Therefore, since S_R and S_Q are sets of pairwise vertex-disjoint paths, we obtain a set S of pairwise vertex-disjoint paths in G connecting each pair in W.

For the second statement, observe that if S is a solution to J, then we can derive a construction of every a-b path in G_Q from Observation 4.3 (analogously to the proof of Lemma 4.5). □

The algorithm. We are now ready to show how to proceed with a given instance $J = (G, T, W)$ of the ComVDP problem. Let $(\mathcal{T}, \mathcal{X})$ be a treecut decomposition of G.

Observe that if t is a leaf of $\mathcal{T}$ and $R \in R(t)$, the corresponding instance (G_R, T_R, W_R) of R is an instance of SComVDP. Indeed, since $Y_t = X_t$, Y_t has at most k elements and the vertices in V_R are of degree at most 2. This means that to compute the set $D(t)$ for a leaf t, for every element R of $R(t)$ we find a corresponding instance and solve SComVDP on it. Since $|R(t)| \leq k^{\mathcal{O}(k)}$, we obtain the following.

Lemma 4.7 *There is an algorithm which takes as input a* ComVDP *instance* (G, T_G, W)*, a treecut decomposition* $(\mathcal{T}, \mathcal{X})$ *of* G *of width* k *and a leaf* $t \in V(\mathcal{T})$*, and computes* $D(t)$ *in time* $k^{\mathcal{O}(k)} \cdot n^2$.

Next, we proceed to the non-leaf nodes. A dynamic programming step will consist of three stages. For a non-leaf node t, we again construct a corresponding instance, but since its size does not have to be bounded by some function of k, we apply some further modifications. First, we apply a reduction rule for each of the thin children (see below). Then we perform a simplification for each bold child t' of t and each $R \in R(t)$. After these, we argue that the graph obtained this way (for a fixed record R) is again an instance of SComVDP, which can be solved efficiently.

Assume we solve an instance (G, T_G, W) and let $t \in V(\mathcal{T})$ be a non-leaf node. The safeness of the following reduction rule follows directly from its definition.

Reduction Rule 1 *Assume that $\mathcal{T}$ is a nice decomposition and let $s \in V(\mathcal{T})$ be a thin child of t. If $D(s)$ is empty, we report a no-instance. Otherwise, we proceed with the first option that applies on the following list. Herein, by* terminating *a terminable set $\mathcal{C}$, we mean to replace G by the result of the termination of $\mathcal{C}$ with respect to $G[Z_s]$. (In particular, this means to remove Y_s from G.)*

1. *If $E_s = \{y_i z_i\}$, and:*
   - *if $((y_i z_i \to L), \emptyset, \emptyset, a \mapsto y_i z_i) \in D(s)$, for some $\{a, b\} \in W$ such that $U_s = \{a\}$, then we terminate $\{\{y_i z_i\}\}$ and we replace $\{a, b\}$ in W with $\{c(\{y_i z_i\}), b\}$.*
   - *if $((y_i z_i \to U), \emptyset, \emptyset, \emptyset) \in D(s)$ and $U_s = \emptyset$, then we remove Y_s from G.*
2. *If $E_s = \{y_i z_i, y_j z_j\}$, $U_s = \emptyset$, and:*
   - *if $((y_i z_i, y_j z_j \to F), \emptyset, \{y_i z_i, y_j z_j\}, \emptyset) \in D(s)$, then we terminate $\mathcal{C} = \{\{y_i z_i, y_j z_j\}\}$.*
   - *if $((y_i z_i, y_j z_j \to U), \emptyset, \emptyset, \emptyset) \in D(s)$, then we remove Y_s from G.*
   - *if $((y_i z_i, y_j z_j \to I), \{y_i z_i, y_j z_j\}, \emptyset, \emptyset) \in D(s)$, then we terminate the pair $\{\{y_i z_i\}, \{y_j z_j\}\}$ and we add $\{c(\{y_i z_i\}), c(\{y_j z_j\})\}$ to W.*
3. *If $E_s = \{y_i z_i, y_j z_j\}$, $U_s = \{a\}$, and $\{a, b\} \in W$, and:*
   - *if $((y_i z_i \to L, y_j z_j \to U), \emptyset, \emptyset, a \mapsto y_i z_i) \in D(s)$ and $((y_j z_j \to L, y_i z_i \to U), \emptyset, \emptyset, a \mapsto y_j z_j) \in D(s)$, then we terminate $\{\{y_i z_i, y_j z_j\}\}$ and we add $\{c(\{y_i z_i, y_j z_j\}), b\}$ to W.*
   - *if $(y_i z_i \to L, y_j z_j \to U), \emptyset, \emptyset, a \mapsto y_i z_i) \in D(s)$, then we terminate $\{\{y_i z_i\}\}$ and we add $\{c(\{y_i z_i\}), b\}$ to W.*
4. *If $E_s = \{y_i z_i, y_j z_j\}$, $U_s = \{a_1, a_2\}$, and $\{a_1, b_1\}, \{a_2, b_2\} \in W$ and:*
   - *if $((y_i z_i, y_j z_j \to L), \emptyset, \emptyset, a_1 \mapsto y_i z_i, a_2 \mapsto y_j z_j) \in D(s)$ and $((y_i z_i, y_j z_j \to L), \emptyset, \emptyset, a_1 \mapsto y_j z_j, a_2 \mapsto y_i z_i) \in D(s)$, then we terminate $\{\{y_i z_i, y_j z_j\}\}$. Let $c = c(\{y_i z_i, y_j z_j\})$. We add to G a twin c' of c, copying the transitions on the incident edges, and we add $\{c, b_1\}$ and $\{c', b_2\}$ to W.*
   - *if $((y_i z_i, y_j z_j \to L), \emptyset, \emptyset, a_1 \mapsto y_i z_i, a_2 \mapsto y_j z_j) \in D(s)$, then we terminate $\{\{y_i z_i\}, \{y_j z_j\}\}$ and add $\{c(\{y_i z_i\}), b_1\}$ and $\{c(\{y_j z_j\}), b_2\}$ to W.*
5. *In all other cases, we report that (G, T, W) is a no-instance.*

□

Finally, we are ready to prove the following.

Lemma 4.8 *There is an algorithm which takes an instance $J = (G, T, W)$ of* ComVDP*, a nice treecut decomposition $(\mathcal{T}, \mathcal{X})$ of G of width k and a non-leaf node $t \in V(\mathcal{T})$, and computes $D(t)$ in time $k^{\mathcal{O}(k)} \cdot n^2$, assuming that for each child t' of t the set $D(t')$ is already computed.*

Proof First, we loop over all possible $R \in R(t)$; recall that $|R(t)| \leq k^{\mathcal{O}(k)}$. For a fixed R, we compute the corresponding instance $J_R = (G_R, T_R, W_R)$ of ComVDP. Recall that V_R is a subset of vertices of G_R, which was added to G_t during the construction of J_R and each vertex in V_R has degree at most two.

For the computed instance J_R, we apply the above reduction rule for each of the thin children of $t \in V(\mathcal{T})$. Note that each vertex v added to G_R during any of these reductions for thin children (denote the set of these vertices by B_B) is of degree at most two. Moreover, since the treecut decomposition of our graph is nice, $N(v) \subseteq X_t \cup V_R$.

Then we loop over all possible functions μ that map each element t' of A_t to some element of $D(t')$. By Theorem 4.5, there are at most $2k + 1$ elements of A_t, so there are at most $(k^{\mathcal{O}(k)})^{2k+1} = k^{\mathcal{O}(k^2)}$ such functions. For the current R and μ, we perform for each $t' \in A_t$ the simplification according to $\mu(t')$. Denote by $\widetilde{J} = (\widetilde{G}, \widetilde{T}, \widetilde{W})$ the instance obtained from J_R after applying this sequence of simplifications (i.e., $\widetilde{G}$ is induced by vertices from $X_t \cup V_R \cup B_B$ and all V_Q's coming from the simplifications).

We claim that $\widetilde{J}$ is an instance of SComVDP. Indeed, we observed that after the simplification process, the degree of a vertex which was not added during the procedure is at most its degree in the original graph. So the only vertices which might have degree bigger than 2 are the ones which belong to X_t. We thus can compute whether $\widetilde{J}$ is a yes-instance in $k^{\mathcal{O}(k)} + O(n^2)$ time using Lemma 4.2.

We claim that $\widetilde{J}$ is a yes-instance if and only if R is valid. To see that, first assume that $R \in R(t)$ is a valid record. Thus J_R admits a solution. By Lemma 4.5, this means that there exists a valid record $R_{t'} \in D(t')$ for each child t' of t. The reduction rule for thin children is safe and thus the solution is preserved. For the bold children, consider an iteration in which μ assigns $R_{t'}$ to t', for each $t' \in A_t$. The second statement of Lemma 4.6 implies that the sequence of simplifications (starting from the instance J_R), consecutively in each node $t' \in A_t$, preserves the existence of a solution, so $\widetilde{J}$ is a yes-instance of SComVDP.

Conversely, assume that the algorithm adds R to $D(t)$, so the instance $\widetilde{J}$ of SComVDP is a yes-instance. Then by the first statement of Lemma 4.6, all the instances obtained in the sequence of simplifications are also yes-instances. Since the reduction rule for thin children is safe, the corresponding instance J_R of R must be also a yes-instance. Therefore, our algorithm computes correctly the set of valid records for t.

The procedure takes time $k^{\mathcal{O}(k^2)}$ times the time needed to solve an instance of SComVDP; together $k^{\mathcal{O}(k^2)} \cdot n^2$. □

We conclude the section with the following theorem.

Theorem 4.6 *There is an algorithm which takes as input a* ComVDP *instance (G, T, W) and returns an answer in time $k^{\mathcal{O}(k^2)} \cdot n^3$.*

Proof We start by computing a treecut decomposition of width $2k$. By Theorem 4.3 this can be done in time $k^{\mathcal{O}(k^2)} \cdot n^2$. Then we use Theorem 4.4 to obtain a nice treecut decomposition in time $\mathcal{O}(n^3)$. We compute the set of valid records for each leaf of $\mathcal{T}$ in time $k^{\mathcal{O}(k^2)} \cdot n^2$ (Lemma 4.7) and then for each non-leaf, by leaf-to-root recursion, in time $k^{\mathcal{O}(k^2)} \cdot n^3$ (Lemma 4.8). We return a positive answer if and only if $D(r) = \{(\emptyset, \emptyset, \emptyset, \emptyset)\}$, where r is the root of $\mathcal{T}$. □

4.3 Edge-Colored Graphs and Treewidth

Our main result on properly colored paths and cycles in edge-colored graphs of bounded treewidth is as follows:

Theorem 4.7 *Given an undirected graph G with an edge coloring $\lambda : E(G) \to [\ell]$ and a tree decomposition $(\mathcal{T}, \beta)$ of G of width less than k, one can verify if G admits a properly colored Hamiltonian Cycle in deterministic time $2^{\mathcal{O}(k)} \cdot \mathcal{O}(|V(G)|+|V(\mathcal{T})|+\ell)$.*

The main highlight of Theorem 4.7 is the lack of the dependency on ℓ in the exponential part of the running time bound (see below as to why this is). For sake of simplicity, we do not analyze in detail the base of the exponent in the running time bound of the algorithm of Theorem 4.7.

The structure of the algorithm of Theorem 4.7 follows the outline of the typical *rank-based* algorithms for connectivity problems on graphs of bounded treewidth [11]. We refer to [15, Chapter 11] for an exposition of the rank-based approach in the case of Steiner Tree and to [53] for a different short exposition of the presented approach for Hamiltonian Cycle.

We start with describing a naive approach and then show how to reduce its complexity.

A *separation* of a graph G is a pair (A, B) of subgraphs of G such that each edge of G belongs to exactly one of the subgraphs A or B. The *order* of the separation (A, B) is $|V(A) \cap V(B)|$. A *partial solution* for a separation (A, B) is a subgraph P of A that is a family of vertex-disjoint paths with both endpoints in $V(A) \cap V(B)$ and such that every vertex of $V(A) \setminus V(B)$ is on one of the paths. Clearly, if C is a Hamiltonian cycle in G and (A, B) is a separation in G with $V(B) \setminus V(A) \neq \emptyset$, then $C \cap A := (V(C) \cap V(A), E(C) \cap E(A))$ is a partial solution for (A, B). If P is a partial solution for (A, B) and Q is a partial solution for (B, A), then we say that P *and* Q *fit each other* if $P \cup Q$ is a Hamiltonian cycle in G.

The *trace* of a partial solution P for (A, B) is a pair (f_P, M_P) where

- $f_P : V(A) \cap V(B) \to \{0, 1, 2\}$ and $f_P(v)$ is the degree of v in P;
- M_P is a matching on the vertex set $f_P^{-1}(1)$, matching endpoints of the paths of P.

Note that the set of possible traces for (A, B) is the same as the set of possible traces for (B, A). Two traces (f_P, M_P) and (f_Q, M_Q) *fit each other* if

- $f_P(v) + f_Q(v) = 2$ for every $v \in V(A) \cap V(B)$; and
- $M_P \uplus M_Q$ is a single cycle on vertex set $f_P^{-1}(1) = f_Q^{-1}(1)$.

The following observation is straightforward.

Lemma 4.9 *If P is a partial solution for (A, B) and Q is the partial solution for (B, A), then P fits Q if and only if the trace of P fits the trace of Q.*

Lemma 4.9 is the base of the naive algorithm for Hamiltonian cycle on graphs of bounded treewidth. Given a tree decomposition $(\mathcal{T}, \beta)$, where $\mathcal{T}$ is a rooted tree, for a node $t \in V(\mathcal{T})$ we use the following notation:

- V_t is the union of $\beta(s)$ over all descendants s of t (including t),
- $\bar{V}_t := (V(G) \setminus V_t) \cup \beta(t)$,
- G_t is the subgraph $G[V_t] \setminus E(G[\beta(t)])$,
- $\bar{G}_t$ is the subgraph $G[\bar{V}_t]$.

Note that $(G_t, \bar{G}_t)$ is a separation. The algorithm, in the bottom-up fashion, computes for every $t \in V(\mathcal{T})$ the family $\mathcal{A}(t)$ of all traces (f, M) for which there exists a partial solution for $(G_t, \bar{G}_t)$ with trace (f, M). If the width of the decomposition is less than k, then the number of possible traces is $2^{\mathcal{O}(k \log k)}$, yielding $2^{\mathcal{O}(k \log k)} \cdot (|V(G)| + |V(\mathcal{T})|)$ running time bound of the algorithm.

This naive algorithm can be easily adjusted to accommodate edge colors. Assume that we are given an edge coloring $\lambda : E(G) \to [\ell]$ and look for a properly colored Hamiltonian cycle. In the definition of a partial solution, we require all paths of P to be properly colored. Furthermore, partial solutions P for (A, B) and Q for (B, A) *fit each other* if their union is a properly colored Hamiltonian cycle, that is, for every vertex $v \in V(A) \cap V(B)$ that is an endpoint of both a path of P and a path of Q, the edge of P incident with v and the edge of Q incident with v are of different colors. To accommodate this, the *colored trace* for a partial solution P is a triple (f_P, M_P, ζ_P) where f_P and M_P have the meaning as before and $\zeta_P : f_P^{-1}(1) \to [\ell]$ assigns to every vertex v of degree 1 in P the color of the unique edge of P incident with v. Then, (f_P, M_P, ζ_P) and (f_Q, M_Q, ζ_Q) fit each other if and only if (f_P, M_P) and (f_Q, M_Q) agree as before and also $\zeta_P(v) \neq \zeta_Q(v)$ for every $v \in f_P^{-1}(1) = f_Q^{-1}(1)$. Again, we have a straightforward analog of Lemma 4.10:

Lemma 4.10 *If P is a partial solution for (A, B) and Q is the partial solution for (B, A) in an edge-colored graph G, then P fits Q if and only if the colored trace of P fits the colored trace of Q.*

Note that there are at most $2^{\mathcal{O}(k \log k)} \ell^k$ possible colored traces for a separation of order at most k. By following the standard dynamic programming algorithm for Hamiltonian cycle on graphs of bounded treewidth (see e.g. [53]), we obtain the following.

Theorem 4.8 *Given an undirected graph G with an edge coloring $\lambda : E(G) \to [\ell]$ and a tree decomposition $(\mathcal{T}, \beta)$ of G of width less than k, one can verify if G admits a properly colored Hamiltonian Cycle in deterministic time $2^{\mathcal{O}(k(\log k + \log \ell))} \cdot \mathcal{O}(|V(G)| + |V(\mathcal{T})| + \ell)$.*

We now introduce the rank-based approach. Fix a separation (A, B) and fix a function $f : V(A) \cap V(B) \to \{0, 1, 2\}$. Let $Z := f^{-1}(1)$; for every trace (f, M), M is a matching on vertex set Z. A *cut* of Z is an unordered pair $\{Z_1, Z_2\}$ such that Z_1

and Z_2 form a partition of Z; a cut *agrees* with a matching M on Z if no edge of M connects the two sides of a cut. Note that there are $2^{|Z|-1}$ cuts.

For two matchings M_1 and M_2 on Z, let $\mathbf{M}[M_1, M_2]$ be 1 if $M_1 \uplus M_2$ is a single cycle and 0 otherwise. For a matching M and a cut C, let $\mathbf{C}[M, C]$ be 1 if M agrees with C and 0 otherwise. The crux of the rank-based approach lies in the following identity (see e.g. Lemma 11.9 of [15]):

Lemma 4.11 *If* $\mathbf{M}$ *and* $\mathbf{C}$ *are treated as matrices over* $\mathbb{F}_2$*, then*

$$\mathbf{M} = \mathbf{C} \cdot \mathbf{C}^T.$$

Let $\mathcal{A}$ be a family of traces for (A, B). We say that $\mathcal{A}' \subseteq \mathcal{A}$ *represents* $\mathcal{A}$ if for every trace τ, if there exists a trace $\tau_1 \in \mathcal{A}$ fitting τ, then there exists a trace $\tau_2 \in \mathcal{A}'$ fitting τ.

Assume that all elements of $\mathcal{A}$ have f as the first coordinate. Let $\mathbf{M}[\mathcal{A}, \cdot]$ be the submatrix of $\mathbf{M}$ induced by the rows of the matchings $\{M \mid (f, M) \in \mathcal{A}\}$. Then, if $\mathcal{A}' \subseteq \mathcal{A}$ is such that $\mathbf{M}[\mathcal{A}', \cdot]$ spans the same subspace as $\mathbf{M}[\mathcal{A}, \cdot]$, then $\mathcal{A}'$ represents $\mathcal{A}$. By the factorization of Lemma 4.11, we infer that:

Lemma 4.12 *Assume that* $\mathcal{A}$ *is a family of traces for* (A, B) *with the same first coordinate* f*. If* $\mathcal{A}' \subseteq \mathcal{A}$ *is such that* $\mathbf{C}[\mathcal{A}', \cdot]$ *spans the same subspace as* $\mathbf{C}[\mathcal{A}, \cdot]$*, then* $\mathcal{A}'$ *represents* $\mathcal{A}$*.*

Consequently, we can improve the naive algorithm for the Hamiltonian Cycle problem as follows. At node $t \in V(\mathcal{T})$, replace $\mathcal{A}(t)$ with a subset $\mathcal{A}'(t)$ representing $\mathcal{A}(t)$ as folows: for every possible $f : \beta(t) \to \{0, 1, 2\}$, restrict $\mathcal{A}(t)$ to $\mathcal{A}(t, f)$ consisting of traces with the first coordinate f, compute $\mathcal{A}'(t, f) \subseteq \mathcal{A}(t, f)$ representing $\mathcal{A}(t, f)$ using Lemma 4.12 and a Gaussian elimination on $\mathbf{C}[\mathcal{A}(t, f), \cdot]$ over $\mathbb{F}_2$, and declare $\mathcal{A}'(t) = \bigcup_f \mathcal{A}'(t, f)$ to be a set representing $\mathcal{A}(t)$. Note that $\mathcal{A}'(t, f)$ is of size at most $2^{|f^{-1}(1)|-1}$ as the number of columns of $\mathbf{C}$ is $2^{|f^{-1}(1)|-1}$; hence $\mathcal{A}'(t)$ is of size at most

$$\sum_{k=0}^{|\beta(t)|} \binom{|\beta(t)|}{k} 2^{k-1} 2^{|\beta(t)|-k} \leq 4^{|\beta(t)|} \leq 4^k.$$

By now-standard methods (see, e.g., the exposition of [53]), one can compute in a bottom-up fashion representatives $\mathcal{A}'(t)$ for $t \in V(\mathcal{T})$; the computation at note t takes into account the representatives at children of t and takes time $2^{\mathcal{O}(k)}$ per child.

By following the same outline, to prove Theorem 4.7, it suffices to show the following:

Lemma 4.13 *Let* G *be a graph with edge coloring* $\lambda : E(G) \to [\ell]$*,* (A, B) *be a separation of order* k*, and* $\mathcal{A}$ *be a family of colored traces for* (A, B)*. Then, there exists a polynomial-time algorithm that, given* $\mathcal{A}$ *and integers* k *and* ℓ*, finds a subset* $\mathcal{A}' \subseteq \mathcal{A}$ *that represents* $\mathcal{A}$ *and is of size at most* 6^k*.*

By partitioning $\mathcal{A}$ according to the first coordinate, it suffices to prove the following:

Lemma 4.14 *Let G be a graph with edge coloring $\lambda : E(G) \to [\ell]$, (A, B) be a separation of order k, $f : V(A) \cap V(B) \to \{0, 1, 2\}$, and $\mathcal{A}$ be a family of colored traces for (A, B) with the first coordinate f. Then, there exists an algorithm that, given $\mathcal{A}$, f, and integers k and ℓ,in time polynomial in the input size and 2^k, finds a subset $\mathcal{A}' \subseteq \mathcal{A}$ that represents $\mathcal{A}$ and is of size at most 4^k.*

The rest of this section is devoted to the proof of Lemma 4.14.

Let a be such that $2^a > \ell$ and let $\mathbb{F} = \mathbb{F}_{2^a}$ be the field of characteristic 2 with 2^a elements. Operations on $\mathbb{F}$ can be done in time polynomial in $a = \mathcal{O}(\log \ell)$. Furthermore, $\mathbb{F}_2 \subseteq \mathbb{F}$. We replace the range of colors $[\ell]$ with non-zero elements of $\mathbb{F}$: for every color $i \in [\ell]$ we pick a distinct non-zero element $a_i \in \mathbb{F}$. Henceforth, we assume that λ and all functions ζ_P for $(f, M_P, \zeta_P) \in \mathcal{A}$ have range $\{a_i \mid i \in [\ell]\} \subseteq \mathbb{F} \setminus \{0\}$.

Let $Z = f^{-1}(1)$. For two functions $\zeta_P, \zeta_Q : Z \to \mathbb{F}$, define

$$\pi(\zeta_P, \zeta_Q) = \prod_{v \in Z} (\zeta_P(v) - \zeta_Q(v)).$$

Note that π can be treated as a $2|Z|$-variate multilinear polynomial of degree $|Z|$ with variables $(\zeta_P(v))_{v \in Z}$ and $(\zeta_Q(v))_{v \in Z}$. Furthermore, we have that

$$\pi(\zeta_P, \zeta_Q) = 0 \quad \Leftrightarrow \quad \exists_{v \in Z} \zeta_P(v) = \zeta_Q(v). \tag{1}$$

For a function $\zeta_P : Z \to \mathbb{F}$, define a $|Z|$-variate multilinear polynomial

$$\pi_{\zeta_P}((x_v)_{v \in Z}) = \pi(\zeta_P, \{v \mapsto x_v \mid v \in Z\}).$$

Consider a matrix $\mathbf{D}$ with rows indexed by possible colored traces (f, M_P, ζ_P) for (A, B) and columns by all $2^{|Z|}$ multilinear monomials on variables $(x_v)_{v \in Z}$. The row $\mathbf{D}[(f, M_P, \zeta_P), \cdot]$ contains the coefficients of π_{ζ_P}.

Let $\mathbf{C}'$ be a matrix with rows indexed by possible colored traces (f, M_P, ζ_P) for (A, B) and columns by all cuts of Z. The row $\mathbf{C}'[(f, M_P, \zeta_P), \cdot]$ equals $\mathbf{C}[(f, M_P), \cdot]$, where every element is treated as an element of $\mathbb{F}$. Finally, let $\mathbf{E}$ be a matrix with rows indexed by possible colored traces τ for (A, B) and $\mathbf{E}[\tau, \cdot]$ is the tensor product of $\mathbf{C}'[\tau, \cdot]$ and $\mathbf{D}[\tau, \cdot]$. Note that $\mathbf{E}$ has $2^{|Z|-1} \cdot 2^{|Z|} = 2^{2|Z|-1}$ columns.

Lemma 4.14 follows from the following lemma by applying Gaussian elimination on the rows of $\mathbf{E}$ corresponding to the elements of $\mathcal{A}$.

Lemma 4.15 *Let $\mathcal{A}' \subseteq \mathcal{A}$ be such that $\mathbf{E}[\mathcal{A}', \cdot]$ spans the same subspace as $\mathbf{E}[\mathcal{A}, \cdot]$. Then, $\mathcal{A}'$ represents $\mathcal{A}$.*

Proof Let $\tau_Q = (f_Q, M_Q, \zeta_Q)$ be a colored trace for (B, A) and let $\tau_P = (f, M_P, \zeta_P) \in \mathcal{A}$ be a trace fitting τ_Q. Note that $f_Q^{-1}(1) = Z$.

Let v_1 be a vector over $\mathbb{F}$ with elements indexed by all cuts of Z with value 1 if the corresponding cut agrees with M_Q and 0 otherwise. By Lemma 4.11, $\mathbf{C}[(f, M), \cdot] \cdot v_1 \neq 0$ if and only if the trace (f, M) fits (f_Q, M_Q).

Let v_2 be a vector over $\mathbb{F}$ with elements indexed by all $2^{|Z|}$ multilinear monomials over variables $(x_v)_{v\in Z}$; the value of v_2 at monomial $\prod_{v\in I} x_v$ for $I \subseteq Z$ equals $\prod_{v\in I} \zeta_Q(v)$. For a colored trace (f, M_R, ζ_R), by (1), we have that $\zeta_R(v) \neq \zeta_Q(v)$ for every $v \in Z$ if and only if $\mathbf{D}[(f, M_R, \zeta_R), \cdot] \cdot v_2 \neq 0$.

Consequently, for a colored trace $\tau = (f, M_R, \zeta_R)$, we have that τ fits τ_Q if and only if $\mathbf{C}'[\tau, \cdot]\cdot v_1 \neq 0$ and $\mathbf{D}[\tau, \cdot]\cdot v_2 \neq 0$. The latter is equivalent to $\mathbf{E}[\tau, \cdot]\cdot(v_1 \otimes v_2) \neq 0$, where $v_1 \otimes v_2$ is the tensor product of v_1 and v_2.

Since τ_P fits τ_Q, $\mathbf{E}[\tau_P, \cdot] \cdot (v_1 \otimes v_2) \neq 0$. Since $\mathbf{E}[\mathcal{A}', \cdot]$ spans the same subspace as $\mathbf{E}[\mathcal{A}, \cdot]$, there exist elements $\tau_1, \tau_2, \ldots, \tau_r \in \mathcal{A}'$ and coefficients $\lambda_1, \ldots, \lambda_r$, such that

$$\mathbf{E}[\tau_P, \cdot] = \sum_{i=1}^{r} \lambda_i \mathbf{E}[\tau_i, \cdot].$$

Since $\mathbf{E}[\tau_P, \cdot] \cdot (v_1 \otimes v_2) \neq 0$, there exists $1 \leq i \leq r$ such that $\lambda_i \neq 0$ and $\mathbf{E}[\tau_P, \cdot] \cdot (v_1 \otimes v_2) \neq 0$. Hence, τ_i fits τ_Q and we have $\tau_i \in \mathcal{A}'$. This finishes the proof of the lemma. □

Recall that $\mathbf{E}$ has $2^{2|Z|-1} \leq 4^{|Z|}$ columns. Hence, with Gaussian elimination one can find $\mathcal{A}'$ as in Lemma 4.15 of size at most $4^{|Z|}$. This finishes the proof of Theorem 4.7.

5 Two Disjoint Shortest Paths

In this section, we present polynomial-time algorithms for vertex-disjoint and edge-disjoint cases of 2- DSPP WITH TRANSITION RESTRICTIONS when every directed cycle has positive length. First, we give the formal definition of this problem.

DIRECTED TWO DISJOINT SHORTEST PATHS PROBLEM (2- DSPP) WITH TRANSITION RESTRICTIONS

Input: A directed graph $G = (V, E)$ with transition system T, a length function $w : E \to \mathbb{R}_{\geq 0}$ and two pairs of vertices (s_1, t_1),(s_2, t_2) in G.

Task: Find two disjoint (vertex-disjoint or edge-disjoint) paths P_1 and P_2 in G such that for both $i = 1, 2$, path P_i is a shortest path (even in the graph G with no transition restrictions) from s_i to t_i and P_i is also T-compatible.

Our algorithm is an adaption of the algorithm for DIRECTED TWO DISJOINT SHORTEST PATHS PROBLEM (assuming that every dicycle in G has positive length) of Bérczi and Kobayashi [9], which reduces the problem of EDGE DISJOINT 2- DSPP to finding a path in a graph $\mathcal{G}$ constructed from the input graph G. Roughly speaking, we show that transition restrictions are not a barrier for using the same strategy. Note that in this section we consider directed graphs with parallel edges. Transitions and transition systems for directed graphs are defined in the natural way analogous to the undirected case.

We follow the notations from the paper of Bérczi and Kobayashi [9] for convenience. We define E_i to be the set of edges that appear in some shortest path (without transition

restrictions) from s_i to t_i for $i = 1, 2$. By this definition, an s_i-t_i path is a shortest T-compatible s_i-t_i path if and only if it consists of edges of E_i and is also T-compatible for $i = 1, 2$. Thus the edge-disjoint (vertex-disjoint) 2- DSPP WITH TRANSITION RESTRICTIONS is equivalent to finding two edge-disjoint (vertex-disjoint) T-compatible paths P_1 and P_2 such that P_i is from s_i to t_i, $E(P_i) \subseteq E_i$ and P_i satisfies the transition restrictions for $i = 1, 2$. Each set E_i can be computed in polynomial time using the method from the paper of Bérczi and Kobayashi [9]. First, we compute the distance $d_i(v)$ from s_i to v for $i = 1, 2$, using Dijkstra's algorithm. Let $\mathcal{E}_i = \{uv \mid d_i(v) - d_i(u) = w(uv)\}$. Then $E_i = \{uv \in \mathcal{E}_i \mid \text{there exists a path from } v \text{ to } t_i \text{ in } \mathcal{E}_i\}$.

Theorem 5.1 *If the length of every directed cycle is positive, both edge-disjoint and vertex-disjoint variants of* 2- DSPP WITH TRANSITION RESTRICTIONS *can be solved in polynomial time.*

Corollary 5.1 *If the length of every edge is positive, both edge-disjoint and vertex-disjoint variants of* 2- DSPP WITH TRANSITION RESTRICTIONS *can be solved in polynomial time.*

For a set F of directed edges, let $\overline{F}$ be the set of edges obtained by reversing all edges of F, that is, $\overline{F} = \{vu \mid uv \in F\}$. For a directed edge $e = uv$, let $\overline{e} = vu$ denote the edge obtained by reversing e. For two paths P and Q with consecutive edges $e^p_1, e^p_2, \ldots, e^p_{|P|}$ and, respectively, $e^q_1, e^q_2, \ldots, e^q_{|Q|}$ such that $\text{head}(e^p_{|P|}) = \text{tail}(e^q_1)$, by $P \cdot Q$ we denote the concatenation of paths P and Q, i.e., $P \cdot Q = e^p_1, e^p_2, \ldots, e^p_{|P|}, e^q_1, e^q_2, \ldots, e^q_{|Q|}$. Note that if P and Q are vertex-disjoint except for $\text{head}(e^p_{|P|}) = \text{tail}(e^q_1)$, then $P \cdot Q$ is a path, too.

5.1 Edge-Disjoint Case

We show that the edge-disjoint case of 2- DSPP WITH TRANSITION RESTRICTIONS can be solved in polynomial time. We use the method of Bérczi and Kobayashi [9], which reduces the problem of EDGE DISJOINT 2- DSPP to finding a path in an auxiliary graph $\mathcal{G}$ suitably constructed from the input graph G. Based on that, we just need to delete edges of $\mathcal{G}$ which correspond to forbidden transitions of G with respect to T and it suffices to find the path in the remaining subgraph of $\mathcal{G}$.

Ordinary graphs. We repeat the procedure of Bérczi and Kobayashi [9] briefly here for consistency (see Fig. 6). Let G be a graph (without transition systems T) such that the length of every dicycle in G is positive. First, we compute E_i for $i = 1, 2$. Then we create four new vertices s'_1, s'_2, t'_1, t'_2, create four edges s'_1s_1, s'_2s_2, $t_1t'_1$, $t_2t'_2$ of length 0 respectively, and add s'_is_i, $t_it'_i$ to E_i for $i = 1, 2$. Let $E_0 = E_1 \cap E_2$, $E^*_1 = E_1 \setminus E_0$, $E^*_2 = E_2 \setminus E_0$. We remove all edges of $E(G) \setminus (E_1 \cup E_2)$, contract all edges of E_0 and reverse all edges of E^*_2. Finally we get a new graph $G^* = (V^*, E^* = E^*_1 \cup \overline{E^*_2})$. Let $V_0 \subseteq V$ be the set of vertices that are newly created after contracting E_0. For $v \in V_0$, we use G_v to denote the subgraph of $G - (E(G) \setminus (E_1 \cup E_2))$ induced by the vertices corresponding to v before contracting. For an edge $e \in E^*$, let $f(e) \in E(G)$ be the edge corresponding to e before the contracting and reversing operations.

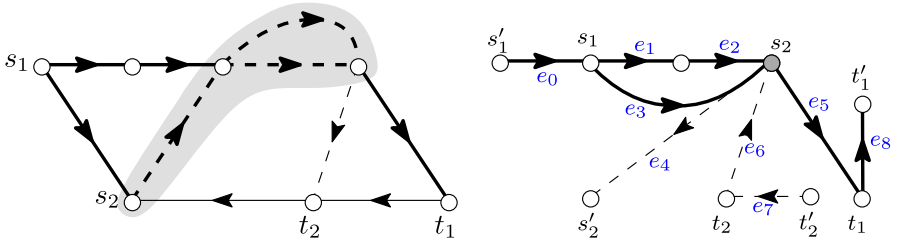

Fig. 6 A digraph G and its corresponding graph G^* (to simplify the picture, we omitted the lengths of edges). Here, edges in E_1 are fat, and the edges of E_2 are dashed. Set V_0 is a singleton, denoted by gray vertex s_2, that corresponds to the gray subgraph G_{s_2} of G

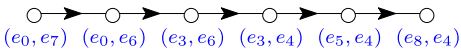

Fig. 7 A path in $\mathcal{G}$ that corresponds to the solution of the instance G depicted in Fig. 6

Note that $s_i, t_i \in V(G)$ might be the endpoints of edges of E_0 for $i = 1, 2$. In this case, although we might contract the edges incident to $s_i, t_i \in V(G)$ and replace these vertices with new vertices. Therefore, we slightly abuse the notation and use s_i and t_i to denote the vertex adjacent to s'_i and t'_i respectively in G^* for $i = 1, 2$, for the sake of simplicity.

The following two lemmas show that G_v is acyclic for every $v \in V_0$ and G^* is acyclic.

Lemma 5.1 *(Bérczi, Kobayashi, [9]) The edge set E_i forms no dicycle in G for $i = 1, 2$.*

Lemma 5.2 *(Bérczi, Kobayashi, [9]) In the graph G, suppose that C is a dicycle in $E_1 \cup \overline{E_2}$. Then $E_1 \cap E(C) \subseteq E_2$ and $E_2 \cap \overline{E(C)} \subseteq E_1$.*

Then we define a new digraph $\mathcal{G}$ whose vertex set is $W = E_1^* \times \overline{E_2^*}$. There is a directed edge from (e_1, e_2) to (e'_1, e'_2) if one of three cases holds.

(i) $e_1 = e'_1$ and $\mathrm{head}_{G^*}(e_2) = \mathrm{tail}_{G^*}(e'_2) = v$. There is no path from $\mathrm{head}_{G^*}(e_1)$ to v in G^*. Moreover, if $v \in V_0$, then G_v contains a path from $\mathrm{tail}_G(e'_2)$ to $\mathrm{head}_G(e_2)$.
(ii) $e_2 = e'_2$ and $\mathrm{head}_{G^*}(e_1) = \mathrm{tail}_{G^*}(e'_1) = v$. There is no path from $\mathrm{head}_{G^*}(e_2)$ to v in G^*. Moreover, if $v \in V_0$, then G_v contains a path from $\mathrm{head}_G(e_1)$ to $\mathrm{tail}_G(e'_1)$.
(iii) $\mathrm{head}_{G^*}(e_2) = \mathrm{tail}_{G^*}(e'_2) = \mathrm{head}_{G^*}(e_1) = \mathrm{tail}_{G^*}(e'_1) = v$. If $v \in V_0$, then G_v contains two edge-disjoint paths from $\mathrm{head}_G(e_1)$ to $\mathrm{tail}_G(e'_1)$ and from $\mathrm{tail}_G(e'_2)$ to $\mathrm{head}_G(e_2)$ respectively.

Finally the following lemma reduces the edge-disjoint version of 2- DSPP to finding a path in $\mathcal{G}$ from $(s'_1 s_1, t'_2 t_2)$ to $(t_1 t'_1, s_2 s'_2)$ (see Fig. 7).

Lemma 5.3 *(Bérczi, Kobayashi, [9]) There is a directed path in $\mathcal{G}$ from $(s'_1 s_1, t'_2 t_2)$ to $(t_1 t'_1, s_2 s'_2)$ if and only if G has two edge-disjoint paths P_1 and P_2 such that P_i is from s'_i to t'_i and $P_i \subseteq E_i$ for $i = 1, 2$.*

Handling transitions. To solve the edge-disjoint version of DIRECTED TWO DISJOINT SHORTEST PATHS PROBLEM (2- DSPP) WITH TRANSITION RESTRICTIONS, we will show that it suffices to (i) delete these edges in $\mathcal{G}$ that correspond to forbidden

transitions (or non-T-compatible paths) of G, and (ii) find a path in the remaining graph of $\mathcal{G}$ from $(s_1's_1, t_2't_2)$ to (t_1t_1', s_2s_2'). For every edge in $\mathcal{G}$, we check whether it corresponds to forbidden transitions according to the following three cases and delete the edge if it corresponds to forbidden transitions. Suppose the edge is from some vertex $(e_1, e_2) \in W$ to another vertex $(e_1', e_2') \in W$.

- The edge is of type (i), i.e., $e_1 = e_1'$ and $\mathrm{head}_{G^*}(e_2) = \mathrm{tail}_{G^*}(e_2') = v$. If $v \in V_0$, let G^s be the subgraph of G consisting of all edges of G_v together with $f(e_2)$ and $f(e_2')$. In this case, if there is no T-compatible paths in G^s from $\mathrm{tail}_G(f(e_2'))$ to $\mathrm{head}_G(f(e_2))$, then remove the edge from $\mathcal{G}$. If $v \notin V_0$ and $\{\overline{e_2'}, \overline{e_2}\} \notin T_G(v)$, then remove the edge from $\mathcal{G}$.
- The edge is of type (ii), i.e., $e_2 = e_2'$ and $\mathrm{head}_{G^*}(e_1) = \mathrm{tail}_{G^*}(e_1') = v$. If $v \in V_0$, let G^s be the subgraph of G consisting of all edges of G_v together with $f(e_1)$ and $f(e_1')$. In this case, if $v \in V_0$ and there is no T-compatible path in G^s from $\mathrm{tail}_G(f(e_1))$ to $\mathrm{head}_G(f(e_1'))$, then remove the edge from $\mathcal{G}$. If $v \notin V_0$ and $\{e_1, e_1'\} \notin T_G(v)$, then remove the edge from $\mathcal{G}$.
- The edge is of type (iii), i.e., $\mathrm{head}_{G^*}(e_2) = \mathrm{tail}_{G^*}(e_2') = \mathrm{head}_{G^*}(e_1) = \mathrm{tail}_{G^*}(e_1') = v$. If $v \in V_0$, let G^s be the subgraph of G consisting of all edges of G_v together with $f(e_1)$, $f(e_1')$, $f(e_2)$ and $f(e_2')$. In this case, if G^s does not contain two T-compatible edge-disjoint paths such that one path is from $\mathrm{tail}_G(f(e_1))$ to $\mathrm{head}_G(f(e_1'))$ and the other path is from $\mathrm{tail}_G(f(e_2'))$ to $\mathrm{head}_G(f(e_2))$, then remove the edge from $\mathcal{G}$. If $v \notin V_0$ and $\{e_1, e_1'\} \notin T_G(v)$ or if $v \notin V_0$ and $\{\overline{e_2'}, \overline{e_2}\} \notin T_G(v)$, then remove the edge from $\mathcal{G}$.

We need to check whether there exists a T-compatible path between two given vertices in a (direced) forbidden-transition graph. Szeider shows a dichotomy of NP-complete and linear-time solvable for the problem of finding a T-compatible path between two given vertices of an (undirected) graph [49]. In contrast, the following lemma shows that in a directed acyclic graph, we can find a T-compatible path between two given vertices in polynomial time.

Lemma 5.4 *In a directed acyclic graph $G = (V, E)$ with transition system T_G, we can compute if there is a directed T-compatible path P from s to t for $s, t \in V(G)$ in polynomial time.*

Proof We construct a directed graph $\tilde{G}$ as follows. First create two vertices s_0, t_0. Then for every edge $e \in E(G)$, create a vertex v_e. For any two edges $e, e' \in E(G)$, create an edge $v_e v_{e'}$ if $ee' \in E(T_G(v))$ for some $v \in V(G)$. Finally, create edges $s_0 v_e$ for every $e \in E(G)$ such that $\mathrm{tail}_G(e) = s$ and create edges $v_{e'} t_0$ for every $e' \in E(G)$ such that $\mathrm{head}_G(e') = t$. We claim that we can find a directed path P' from s_0 to t_0 in $\tilde{G}$ if and only if there is a directed T-compatible path P from s to t in G. For the "if" direction, suppose that there is such a path $P = e_1, e_2, \ldots, e_\ell$ in G, where $e_1, \ldots, e_\ell$ are the consecutive edges of P. Then we can obviously get the path $P' = s_0 v_{e_1}, v_{e_1} v_{e_2}, \ldots, v_{e_\ell} t_0$ by the definition of $\tilde{G}$. For the "only if" direction, suppose that there is a directed path $P' = s_0 v_{e_{i_1}}, v_{e_{i_1}} v_{e_{i_2}}, \ldots, v_{e_{i_\ell}} t_0$ in $\tilde{G}$. Then $P = e_{i_1}, e_{i_2}, \ldots, e_{i_\ell}$ is a directed T-compatible walk from s to t in G. Since G is acyclic, P is also a path. This completes the proof of the claim. We can build the graph

$\tilde{G}$ in $O(|E|^2)$-time and find an s_0t_0 path in $\tilde{G}$ using DFS in $O(|E|^2)$ time. Thus the lemma holds. □

For $v \in V_0$, by Lemma 5.1, there is no dicycle in G_v. Moreover, observe that we cannot have a vertex in $V(G) \setminus V(G_v)$ adjacent to more than one edge from $E(G^s) \setminus E(G_v)$, so G^s is also acyclic. So we can decide whether or not to remove the edges of type (i) or (ii) from $\mathcal{G}$ in polynomial time according to Lemma 5.4. For the edges of type (iii), we need to compute if there are two edge-disjoint T-compatible paths in a directed acyclic graph. We show that it can be done in polynomial time and the algorithm is an adaption of the algorithm of finding two vertex-disjoint paths in DAG by Perl and Shiloach [47].

Lemma 5.5 *In a directed acyclic graph $G = (V, E)$ with transition system T_G, we can solve the edge-disjoint version of* 2- DSPP WITH TRANSITION RESTRICTIONS *in polynomial time.*

Proof First we modify the graph G as follows. We create four vertices s_1', s_2', t_1', t_2' and update $V(G)$ as $V(G) \leftarrow V(G) \cup \{s_1', s_2', t_1', t_2'\}$. We create four edges $s_1's_1, s_2's_2, t_1t_1', t_2t_2'$ and update $E(G)$ as $E(G) \leftarrow E(G) \cup \{s_1's_1, s_2's_2, t_1t_1', t_2t_2'\}$. Also, for $i = 1, 2$, we update $T_G(s_i)$ as

$$T_G(s_i) \leftarrow T_G(s_i) \cup \{\{e, e'\} \mid e = s_i's_i \text{ and } \mathrm{tail}_G(e') = s_i\},$$

and we update $T_G(t_i)$ as

$$T_G(t_i) \leftarrow T_G(t_i) \cup \{\{e, e'\} \mid e' = t_it_i' \text{ and } \mathrm{head}_G(e) = t_i\}.$$

For every vertex $v \in V(G)$, define the level $\ell(v)$ as the length of a longest directed path in G starting from v. Since G is acyclic, this can be computed by repeatedly removing a vertex of G. Then we create a graph $\tilde{G}$ as follows. Let the vertex set of $\tilde{G}$ be $V(\tilde{G}) = \{(e_1, e_2) \mid e_1, e_2 \in E(G) \text{ and } e_1 \neq e_2\}$. For every $(e_1, e_2), (e_1', e_2') \in V(\tilde{G})$, create an edge from (e_1, e_2) to (e_1', e_2') if one of the following cases holds:

(1) $e_1 = e_1'$, $\ell(\mathrm{head}_G(e_2)) \geq \ell(\mathrm{head}_G(e_1))$, $\{e_2, e_2'\} \in T_G(\mathrm{head}_G(e_2))$.
(2) $e_2 = e_2'$, $\ell(\mathrm{head}_G(e_1)) \geq \ell(\mathrm{head}_G(e_2))$, $\{e_1, e_1'\} \in T_G(\mathrm{head}_G(e_1))$.
(3) $e_1 = e_1' = t_1t_1'$, $\ell(\mathrm{head}_G(e_2)) < \ell(t_1')$, $\{e_2, e_2'\} \in T_G(\mathrm{head}_G(e_2))$.
(4) $e_2 = e_2' = t_2t_2'$, $\ell(\mathrm{head}_G(e_1)) < \ell(t_2')$, $\{e_1, e_1'\} \in T_G(\mathrm{head}_G(e_1))$.

We claim that there are two T-compatible edge-disjoint paths P_1 and P_2 in G such that P_i is from s_i' to t_i' for $i = 1, 2$ if and only if there is a path P from $(s_1's_1, s_2's_2)$ to (t_1t_1', t_2t_2') in $\tilde{G}$.

("only if" direction): Let $P_1 = e_1^0, e_1^1, \ldots, e_1^{p+1}$ and $e_1^0 = s_1's_1, e_1^{p+1} = t_1t_1'$. Let $P_2 = e_2^0, e_2^1, \ldots, e_2^{q+1}$ and $e_2^0 = s_2's_2, e_2^{q+1} = t_2t_2'$. For any $i \in \{0, 1, \ldots, p+1\}$, $j \in \{0, 1, \ldots, q+1\}$ such that $(i, j) \neq (p+1, q+1)$, one of the following four cases must hold.

- $i \leq p$ and $j \leq q$, $\ell(\mathrm{head}_G(e_1^i)) \leq \ell(\mathrm{head}_G(e_2^j))$ and there is an edge in $\tilde{G}$ from (e_1^i, e_2^j) to (e_1^i, e_2^{j+1}).

- $i \leq p$ and $j \leq q$, $\ell(\mathrm{head}_G(e_1^i)) \geq \ell(\mathrm{head}_G(e_2^j))$ and there is an edge in $\tilde{G}$ from (e_1^i, e_2^j) to (e_1^{i+1}, e_2^j).
- $i = p+1$ and $j \leq q$, $\ell(\mathrm{head}_G(e_2^j)) < \ell(t_1')$ and there is an edge in $\tilde{G}$ from (e_1^{p+1}, e_2^j) to (e_1^{p+1}, e_2^{j+1}).
- $j = q+1$ and $i \leq p$, $\ell(\mathrm{head}_G(e_1^i)) < \ell(t_2')$ and there is an edge in $\tilde{G}$ from (e_1^i, e_2^{q+1}) to (e_1^{i+1}, e_2^{q+1}).

As a result, there is a path P from $(s_1's_1, s_2's_2)$ to (t_1t_1', t_2t_2') in $\tilde{G}$. This finishes the proof for "only if" direction.

("if" direction): Suppose that there exists a path P from $(s_1's_1, s_2's_2)$ to (t_1t_1', t_2t_2') in $\tilde{G}$. Let $P = (e_1^0, e_2^0), (e_1^1, e_2^1), \ldots, (e_1^r, e_2^r)$ such that $(s_1's_1, s_2's_2) = (e_1^0, e_2^0)$ and $(e_1^r, e_2^r) = (t_1t_1', t_2t_2')$. We construct two edge-disjoint T-compatible paths P_1, P_2 as follows. First we initialize $P_1 = e_1^0$, $P_2 = e_2^0$. Then for $i = 0, \ldots, r-1$, we update P_1 and P_2 according to the following cases:

- Suppose that the edge from (e_1^i, e_2^i) to (e_1^{i+1}, e_2^{i+1}) is of type (1). Then $P_2 \leftarrow P_2 \cdot e_2^{i+1}$.
- Suppose that the edge from (e_1^i, e_2^i) to (e_1^{i+1}, e_2^{i+1}) is of type (2). Then $P_1 \leftarrow P_1 \cdot e_1^{i+1}$.
- Suppose that the edge from (e_1^i, e_2^i) to (e_1^{i+1}, e_2^{i+1}) is of type (3). Then $P_2 \leftarrow P_2 \cdot e_2^{i+1}$.
- Suppose that the edge from (e_1^i, e_2^i) to (e_1^{i+1}, e_2^{i+1}) is of type (4). Then $P_1 \leftarrow P_1 \cdot e_1^{i+1}$.

By the definition of edges of $\tilde{G}$, we get that P_1 and P_2 are two T-compatible edge-disjoint paths in G such that P_i is from s_i' to t_i' for $i = 1, 2$. We can construct a graph $\tilde{G}$ in $O(|E|^3)$ time and find a path from $(s_1's_1, s_2's_2)$ to (t_1t_1', t_2t_2') in $O(|E|^3)$ time. Thus the lemma holds. □

Thus we can also decide whether or not to remove an edge of type (iii) from $\mathcal{G}$ in polynomial time and let $\hat{\mathcal{G}}$ be the remaining subgraph of $\mathcal{G}$. The following lemma shows that we can reduce edge-disjoint version of 2- DSPP WITH TRANSITION RESTRICTIONS to finding a path from $(s_1's_1, t_2't_2)$ to (t_1t_1', s_2s_2') in $\hat{\mathcal{G}}$.

Lemma 5.6 *There is a directed path in $\hat{\mathcal{G}}$ from $(s_1's_1, t_2't_2)$ to (t_1t_1', s_2s_2') if and only if G has two edge-disjoint T-compatible paths P_1 and P_2 such that P_i is from s_i' to t_i' and $P_i \subseteq E_i$ for $i = 1, 2$.*

Proof ("if" direction) Suppose that G has two edge-disjoint T-compatible paths P_1 and P_2 such that P_i is from s_i' to t_i' and $P_i \subseteq E_i$ for $i = 1, 2$. $E(P_1) \setminus E_0$ forms a directed path P_1^* in G^* from s_1' to t_1'. $\overline{E(P_2) \setminus E_0}$ forms a directed path P_2^* in G^* from t_2' to s_2'. Let $P_1^* = e_1^0, e_1^1, ..., e_1^{p+1}$ and $e_1^0 = s_1's_1, e_1^{p+1} = t_1t_1'$. Let $P_2^* = e_2^0, e_2^1, ..., e_2^{q+1}$ and $e_2^0 = t_2't_2, e_2^{q+1} = s_2s_2'$. It follows that $e_1^i \in E_1^*$ for $i = 0, 1, ..., p+1$ and $e_2^j \in E_2^*$ for $j = 0, 1, ..., q+1$. By the proof of Lemma 5.3 (interested readers could refer to the proof of Lemma 8 in [9]), there is a directed path P in $\mathcal{G}$ from

(s'_1s_1, t'_2t_2) to $(t_1t'_1, s_2s'_2)$ such that every edge of P is of one of the three types: (i) from (e_1^i, e_2^j) to (e_1^i, e_2^{j+1}) $(i \in \{0, ..., p+1\}, j \in \{0, ..., q\})$; (ii) from (e_1^i, e_2^j) to (e_1^{i+1}, e_2^j) $(i \in \{0, ..., p\}, j \in \{0, ..., q+1\})$; (iii) from (e_1^i, e_2^j) to (e_1^{i+1}, e_2^{j+1}) $((i \in \{0, ..., p\}, j \in \{0, ..., q\}))$. Since P_1 and P_2 are T-compatible, by the rules we construct $\hat{\mathcal{G}}$, we can see that all edges of P in $\mathcal{G}$ remains in $\hat{\mathcal{G}}$. This completes the proof for "if direction".

("only if" direction) Suppose that there is a directed path P from $(e_1^0, e_2^0) = (s'_1s_1, t'_2t_2)$ to $(e_1^r, e_2^r) = (t_1t'_1, s_2s'_2)$ in $\hat{\mathcal{G}}$ that goes through (e_1^0, e_2^0), (e_1^1, e_2^1), ..., (e_1^r, e_2^r) consecutively. Since $\hat{\mathcal{G}}$ is a subgraph of $\mathcal{G}$, by Lemma 5.3, there exists two edge-disjoint paths P_1 and P_2 in G such that P_i is from s'_i to t'_i and $P_i \subseteq E_i$ for $i = 1, 2$ in G. Moreover, again from the proof of Lemma 5.3, it follows that $e_1^i \in E(P_1)$ and $\overline{e_2^i} \in E(P_2)$. By the rule we construct $\hat{\mathcal{G}}$, for an edge from (e_1^i, e_2^i) to (e_1^{i+1}, e_2^{i+1}) $(i \in \{0, ..., r-1\})$, there is a T-compatible subpath of P_1 from $\mathrm{tail}_G(f(e_1^i))$ to $\mathrm{head}_G(f(e_1^{i+1}))$ if $e_1^i \neq e_1^{i+1}$ or there is a T-compatible subpath of P_2 from $\mathrm{tail}_G(f(e_2^{i+1}))$ to $\mathrm{head}_G(f(e_2^i))$ if $e_2^i \neq e_2^{i+1}$. It follows that P_1 and P_2 are also T-compatible. This finishes the proof for "only if" direction. □

Since $\hat{\mathcal{G}}$ is a subgraph of $\mathcal{G}$ and $\mathcal{G}$ contains at most $|E|^2$ vertices, we can detect a path in $\hat{\mathcal{G}}$ in polynomial time. Thus Lemma 5.6 shows that we can solve edge-disjoint version of 2- DSPP WITH TRANSITION RESTRICTIONS in polynomial time assuming that every cycle in the input graph has positive length.

5.2 Vertex-Disjoint Case

When computing vertex-disjoint version of 2- DSPP in the paper of Bérczi and Kobayashi [9], they create a new digraph G_2 as follows: for every vertex $v \in V$ create two vertices v^+ and v^-. Create an edge v^-v^+ with $w(v^-v^+) = 0$. Create an edge u^+v^- if there is an edge uv in G and let $w(u^+v^-) = w(uv)$. Thus VERTEX- DISJOINT 2- DSPP in G is reduced to edge-disjoint variant of 2- DSPP in G_2. However, this method does not work in the forbidden-transitions setting because part of the information of transitions will be lost after creating the new graph G_2.

In order to keep the information of transitions, we first modify G as follows. We compute the set E_1 and E_2 of G. Remove all edges of $E(G) \setminus (E_1 \cup E_2)$ from $E(G)$ and all isolated vertices from $V(G)$. When removing the edges or vertices we update the transition system accordingly. Then create four new vertices s'_1, s'_2, t'_1, t'_2 and four edges s'_1s_1, s'_2s_2, $t_1t'_1$, $t_2t'_2$ all with length 0. Add s'_is_i and $t_it'_i$ to E_i for $i = 1, 2$. Thus a shortest path from s_i to t_i corresponds to a shortest path from s'_i to t'_i starting with the edge s'_is_i and ending with the edge $t_it'_i$. We update $T_G(s_i)$ by adding $\{\{e, e'\} \mid e = s'_is_i \text{ and } \mathrm{tail}_G(e') = s_i\}$ to it for $i = 1, 2$. Let $T_G(t_i) = \{\{e, e'\} \mid \mathrm{head}_G(e) = t_i \text{ and } e' = t_it'_i\}$ for $i = 1, 2$.

Then we create a graph G' as follows. For every vertex $v \in V(G) \setminus \{s'_1, s'_2, t'_1, t'_2\}$, create two vertices v^+ and v^-. We also create four vertices s'_1, s'_2, t'_1, t'_2 in G' and create four edges $s'_1s_1^-$, $s'_2s_2^-$, $t_1^+t'_1$, $t_2^+t'_2$ in G' all with length 0. For every vertex $v \in V(G) \setminus \{s'_1, s'_2, t'_1, t'_2\}$, let $in_1(v), \ldots, in_{r_v}(v)$ be the incoming edges of v. Then create

r_v parallel edges $e_1(v), \ldots, e_{r_v}(v)$ with $\text{tail}_{G'}(e_j(v)) = v^-$ and $\text{head}_{G'}(e_j(v)) = v^+$ in G' for $j = 1, \ldots, r_v$ such that each of the edges is of length 0. If there is an edge $uv = in_p(v)$ in G for some $p \in [r_v]$ and $u, v \notin \{s'_1, s'_2, t'_1, t'_2\}$, create an edge $in_p(v^-) = u^+v^-$ in G' and let $w(u^+v^-) = w(uv)$. Next, we define the transition system for G' as follows. $T_{G'}(v^-) = \{\{in_j(v^-), e_j(v)\} \mid j \in [r_v]\}$. For every $e, e' \in (E_1 \cup E_2) \setminus \{t_1t'_1, t_2t'_2\} \subseteq E(G)$ such that $e = uv = in_p(v)$, $e' = vw$ (let $\hat{e} = v^+w^-$), if $\{e, e'\} \in T_G(v)$, then $\{e_p(v), \hat{e}\} \in T_{G'}(v^+)$. In particular, let $e_i = t_i^+t'_i$ for $i = 1, 2$. If $e = ut_i = in_q(t_i) \in E(G)$ for some $q \in [r_{t_i}]$, then $\{e_q(t_i), e_i\} \in T_{G'}(t_i^+)$.

We also need to compute the set of edges E'_i that exist in some shortest path (without transitions) from s'_i to t'_i for $i = 1, 2$. By this definition, obviously $s'_is_i^-, s_i^-s_i^+, t_i^-t_i^+, t_i^+t'_i \in E'_i$ for $i = 1, 2$.

Lemma 5.7 *For $u, v \in V(G) \setminus \{s'_1, s'_2, t'_1, t'_2\}$, $uv \in E_i$ if and only if $u^+v^- \in E'_i$ for $i = 1, 2$. Moreover, if some incoming edge of v^- belongs to E'_i, then all of the parallel edges v^-v^+ belong to E_i for $i = 1, 2$.*

Proof Suppose that $P_1 = s_1, w, ..., u, v, ..., t_1$ is a shortest path from s_1 to t_1 in G. We claim that $P'_1 = s'_1, s_1^-, s_1^+, w^-, w^+, ..., u^-, u^+, v^-, v^+, ..., t_1^-, t_1^+, t'_1$ is a shortest path from s'_1 to t'_1 in G'. For contradiction, suppose the claim is not true. Then we can find a path $P'_0 = s'_1, s_1^-, s_1^+, w_1^-, w_1^+, ..., w_\ell^-, w_\ell^+, t_1^-, t_1^+, t'_1$ in G' such that $w(P'_0) < w(P'_1) = w(P_1)$. Then there is a path $P_0 = s_1, w_1, ..., w_\ell, t_1$ in G such that $w(P_0) = w(P'_0) < w(P_1)$, contradicting that P_1 is a shortest path from s_1 to t_1.

Suppose that $P'_1 = s'_1, s_1^-, s_1^+, w^-, w^+, ..., u^-, u+, v^-, v^+, ..., t_1^-, t_1^+, t'_1$ is a shortest path from s'_1 to t'_1 in G'. We claim that $P_1 = s_1, w, ..., u, v, ..., t_1$ is a shortest path from s_1 to t_1 in G. For contradiction, suppose that the claim is not true. Then there exists a path $P_0 = s_1w_1...w_\ell t_1$ in G such that $w(P_0) < w(P_1) = w(P'_1)$. Thus there is a path

$$P'_0 = s'_1, s_1^-, s_1^+, w_1^-, w_1^+, ..., w_\ell^-, w_\ell^+, t_1^-, t_1^+, t'_1$$

in G' such that $w(P'_0) = w(P_0) < w(P'_1)$, contradicting that P'_1 is a shortest path from s'_1 to t'_1 in G'.

Similarly we can show that $P_2 = s_2, w, ..., u, v, ..., t_2$ is a shortest path from s_2 to t_2 in G if and only if

$$P'_2 = s'_2, s_2^-, s_2^+, w^-, w^+, ..., u^-, u^+, v^-, v^+, ..., t_2^-, t_2^+, t'_2$$

is a shortest path from s'_2 to t'_2 in G'. It follows that for $u, v \in V(G) \setminus \{s'_1, s'_2, t'_1, t'_2\}$, $uv \in E_i$ if and only if $u^+v^- \in E'_i$ for $i = 1, 2$.

For $i = 1, 2$, as $w(v^-v^+) = 0$, we have that $d_i(v^+) = d_i(v^-) + w(v^-v^+)$. Since some ingoing edge of v^- belongs to E'_i, there is a $v^-t'_i$ path in E'_i. It follows that there is also a $v^+t'_i$ path in E'_i. By the definition of E'_i, all of the parallel edges v^-v^+ belong to E'_i. □

It's not hard to verify that Lemma 5.1 and Lemma 5.2 also apply to G', but we will also state them here for clarity.

Lemma 5.8 *(Bérczi, Kobayashi, [9]) The edge set E'_i forms no dicycle in G' for $i = 1, 2$.*

Lemma 5.9 *(Bérczi, Kobayashi, [9]) In the graph G', suppose that C is a dicycle in $E'_1 \cup \overline{E'_2}$. Then $E'_1 \cap E(C) \subseteq E'_2$ and $E'_2 \cap \overline{E(C)} \subseteq E'_1$.*

Let $E'_0 = E'_1 \cap E'_2$, $E^*_1 = E'_1 \setminus E'_0$, $E^*_2 = E'_2 \setminus E'_0$. We contract all edges of E'_0 and get a graph $G'' = (V'', E'')$. For an edge $e \in E''$, let $f(e) \in E(G')$ denote the edge corresponding to e before the contracting operations. We need to compute the new transition system of G'' as follows. Let $V'_0 \subseteq V''$ be the set of vertices that are newly created after contracting E'_0. For $v \in V'_0$, we use G'_v to denote the subgraph of $G' - (E(G') \setminus (E'_1 \cup E'_2))$ induced by the vertices corresponding to v before contracting. For every $u \in V(G'') \setminus V'_0$, if $f(e)f(e') \in T_{G'}(u)$ then $\{e, e'\} \in T_{G''}(u)$. Let $v \in V'_0$ and $\mathrm{head}_{G''}(e) = \mathrm{tail}_{G''}(e') = v$. If there is a T-compatible path in the subgraph of G' consisting of all edges of G'_v together with $f(e)$ and $f(e')$ from $\mathrm{tail}_{G'}(f(e))$ to $\mathrm{head}_{G'}(f(e'))$, then $\{e, e'\} \in T_{G''}(v)$. By Lemma 5.8, there is no dicycle in G'_v. Moreover, the subgraph of G' consisting of all edges of G'_v together with $f(e)$ and $f(e')$ is also acyclic. So we can compute $T_{G''}(v)$ for every $v \in V'_0$ in polynomial time according to Lemma 5.4. Since $E^*_1 \cap E^*_2 = \emptyset$, then we can reverse all edges of E^*_2 (the lengths of edges unchanged) with E^*_1 unchanged. We get a new graph $G^* = (V^*, E^*)$, such that $V^* = V''$ and $E^* = E^*_1 \cup \overline{E^*_2}$.

Then we also need to compute the new transition systems of G^*. If $e, g \in E^*_1$ and $\{e, g\} \in T_{G''}(v)$ for some $v \in V''$, then $\{e, g\} \in T_{G^*}(v)$. If $e, g \in E^*_2$ and $\{e, g\} \in T_{G''}(v)$ for some $v \in V''$, then $\{\bar{g}, \bar{e}\} \in T_{G^*}(v)$. Here we use $\bar{e}, \bar{g} \in \overline{E^*_2}$ to denote the reverse of e, g respectively.

Claim After reversing the edges of E^*_2, there is no dicycle in G^*.

Proof (of claim) Suppose for contradiction that there is a dicycle C in G^*. By Lemma 5.8, $E(C) \not\subseteq E^*_1, E(C) \not\subseteq \overline{E^*_2}$. It follows that $E(C) \cap E^*_1 \neq \emptyset$ and $E(C) \cap \overline{E^*_2} \neq \emptyset$. Then by Lemma 5.9, $E(C)$ should have been contracted in G'', contradicting that C is a dicycle in G^*. □

We define a new digraph $\mathcal{G}$ as follows. Let $W = E^*_1 \times \overline{E^*_2}$ be its vertex set. For $(e_1, e_2), (e'_1, e'_2) \in W$, there is a directed edge from (e_1, e_2) to (e'_1, e'_2) if one of three cases hold.

(i) $e_1 = e'_1$, $\mathrm{head}_{G^*}(e_2) = \mathrm{tail}_{G^*}(e'_2) = v$ and $\{e_2, e'_2\} \in T_{G^*}(v)$. There is no path from $\mathrm{head}_{G^*}(e_1)$ to v in G^*.

(ii) $e_2 = e'_2$, $\mathrm{head}_{G^*}(e_1) = \mathrm{tail}_{G^*}(e'_1) = v$ and $\{e_1, e'_1\} \in T_{G^*}(v)$. There is no path from $\mathrm{head}_{G^*}(e_2)$ to v in G^*.

(iii) $\mathrm{head}_{G^*}(e_2) = \mathrm{tail}_{G^*}(e'_2) = \mathrm{head}_{G^*}(e_1) = \mathrm{tail}_{G^*}(e'_1) = v$ and both $\{e_1, e'_1\}$ and $\{e_2, e'_2\}$ are in $T_{G^*}(v)$. Furthermore, if $v \in V_0$, let G^s be the subgraph of G' consisting of all edges of G'_v together with $f(e_1)$, $f(e'_1)$, $f(\overline{e_2})$ and $f(\overline{e'_2})$. Then G^s contains two T-compatible vertex-disjoint paths such that one path is from $\mathrm{tail}_{G'}(f(e_1))$ to $\mathrm{head}_{G'}(f(e'_1))$ and the other path is from $\mathrm{tail}_{G'}(f(\overline{e'_2}))$ to $\mathrm{head}_{G'}(f(\overline{e_2}))$.

In the third case above, we claim that v must belong to V_0'. Suppose for contradiction that $v \notin V_0'$. Clearly, $v \notin \{s_1', s_2', t_1', t_2'\}$, as in must be both, head and tail of some edges. So there are two remaining cases. The first case is that $v = u^-$ for some $u \in V(G)$. Then all outgoing edges of u^- in G'' are parallel edges, that is, $\text{head}_{G''}(\overline{e_2}) = \text{head}_{G''}(e_1')$. Then e_1' and e_2 form a cycle in G^*, contradicting that G^* is acyclic. The second case is that $v = u^+$ for some $u \in V(G)$. Then all ingoing edges of u^+ in G'' are parallel edges, that is, $\text{tail}_{G''}(\overline{e_2'}) = \text{tail}_{G''}(e_1)$. Then e_1 and e_2' form a cycle in G^*, contradicting that G^* is acyclic. Thus v must belong to V_0'. Then we need to solve the vertex-disjoint version of 2- DSPP WITH TRANSITION RESTRICTIONS in the acyclic graph $G_v' \cup \{e_1, e_1', \overline{e_2}, \overline{e_2'}\}$. The following lemma shows that we can do it in polynomial time. The algorithm is an adaption of the algorithm of finding two vertex-disjoint paths in DAG given by Perl and Shiloach [47].

Lemma 5.10 *In a directed acyclic graph $G = (V, E)$ with transition system T_G, we can solve the vertex-disjoint version of* 2- DSPP WITH TRANSITION RESTRICTIONS *in polynomial time.*

Proof First we modify the graph G as follows. We create four vertices s_1', s_2', t_1', t_2' and update $V(G)$ as $V(G) \leftarrow V(G) \cup \{s_1', s_2', t_1', t_2'\}$. We create four edges $s_1's_1, s_2's_2, t_1t_1', t_2t_2'$ and update $E(G)$ as $E(G) \leftarrow E(G) \cup \{s_1's_1, s_2's_2, t_1t_1', t_2t_2'\}$. Also, for $i = 1, 2$, we update $T_G(s_i)$ as

$$T_G(s_i) \leftarrow T_G(s_i) \cup \{\{e, e'\} \mid e = s_i's_i \text{ and } \text{tail}_G(e') = s_i\},$$

and we update $T_G(t_i)$ as

$$T_G(t_i) \leftarrow T_G(t_i) \cup \{\{e, e'\} \mid e' = t_it_i' \text{ and } \text{head}_G(e) = t_i\}.$$

For every vertex $v \in V(G)$, define the level $\ell(v)$ as the length of a longest directed path in G starting from v. This can be computed by repeatedly removing a vertex of G. Then we create a graph $\tilde{G}$ as follows. Let the vertex set of $\tilde{G}$ be $V(\tilde{G}) = \{(e_1, e_2) \mid e_1, e_2 \in E(G) \text{ and } e_1 \neq e_2\}$. For every $(e_1, e_2), (e_1', e_2') \in V(\tilde{G})$, create an edge from (e_1, e_2) to (e_1', e_2') if one of the following cases holds:

(1) $e_1 = e_1'$, $\ell(\text{head}_G(e_2)) \geq \ell(\text{head}_G(e_1))$, $\{e_2, e_2'\} \in T_G(\text{head}_G(e_2))$, $\text{head}_G(e_2') \neq \text{tail}_G(e_1)$ and $\text{head}_G(e_2') \neq \text{head}_G(e_1)$.
(2) $e_2 = e_2'$, $\ell(\text{head}_G(e_1)) \geq \ell(\text{head}_G(e_2))$, $\{e_1, e_1'\} \in T_G(\text{head}_G(e_1))$, $\text{head}_G(e_1') \neq \text{tail}_G(e_2)$ and $\text{head}_G(e_1') \neq \text{head}_G(e_2)$.
(3) $e_1 = e_1' = t_1t_1'$, $\ell(\text{head}_G(e_2)) < \ell(t_1')$, $\{e_2, e_2'\} \in T_G(\text{head}_G(e_2))$.
(4) $e_2 = e_2' = t_2t_2'$, $\ell(\text{head}_G(e_1)) < \ell(t_2')$, $\{e_1, e_1'\} \in T_G(\text{head}_G(e_1))$.

We claim that there are two T-compatible vertex-disjoint paths P_1 and P_2 in G such that P_i is from s_i' to t_i' for $i = 1, 2$ if and only if there is a path P from $(s_1's_1, s_2's_2)$ to (t_1t_1', t_2t_2') in $\tilde{G}$.

("only if" direction): Let $P_1 = e_1^0, e_1^1, \ldots, e_1^{p+1}$ and $e_1^0 = s_1's_1, e_1^{p+1} = t_1t_1'$. Let $P_2 = e_2^0, e_2^1, \ldots, e_2^{q+1}$ and $e_2^0 = s_2's_2, e_2^{q+1} = t_2t_2'$. For any $i \in \{0, 1, \ldots, p+1\}$, $j \in \{0, 1, \ldots, q+1\}$, such that $(i, j) \neq (p+1, q+1)$, one of the following four cases must hold.

- $i \leq p$ and $j \leq q$, $\ell(\text{head}_G(e_1^i)) \leq \ell(\text{head}_G(e_2^j))$, then there is an edge in $\tilde{G}$ from (e_1^i, e_2^j) to (e_1^i, e_2^{j+1}).
- $i \leq p$ and $j \leq q$, $\ell(\text{head}_G(e_1^i)) \geq \ell(\text{head}_G(e_2^j))$, then there is an edge in $\tilde{G}$ from (e_1^i, e_2^j) to (e_1^{i+1}, e_2^j).
- $i = p + 1$ and $j \leq q$, $\ell(\text{head}_G(e_2^j)) < \ell(t_1')$, then there is an edge in $\tilde{G}$ from (e_1^{p+1}, e_2^j) to (e_1^{p+1}, e_2^{j+1}).
- $j = q + 1$ and $i \leq p$, $\ell(\text{head}_G(e_1^i)) < \ell(t_2')$, then there is an edge in $\tilde{G}$ from (e_1^i, e_2^{q+1}) to (e_1^{i+1}, e_2^{q+1}).

As a result, there is a path P from $(s_1's_1, s_2's_2)$ to (t_1t_1', t_2t_2') in $\tilde{G}$. This finishes the proof for "only if" direction.
("if" direction): Suppose that there exists a path P from $(s_1's_1, s_2's_2)$ to (t_1t_1', t_2t_2') in $\tilde{G}$. Let $P = (e_1^0, e_2^0), (e_1^1, e_2^1), \ldots, (e_1^r, e_2^r)$, such that $(e_1^0, e_2^0) = (s_1's_1, s_2's_2)$ and $(e_1^r, e_2^r) = (t_1t_1', t_2t_2')$. We construct two vertex-disjoint T-compatible paths P_1, P_2 as follows. First we initialize $P_1 = e_1^0$, $P_2 = e_2^0$. Then for $i = 0, \ldots, r-1$, we update P_1 and P_2 according to the following cases:

- Suppose that the edge from (e_1^i, e_2^i) to (e_1^{i+1}, e_2^{i+1}) is of type (1). Then $P_2 \leftarrow P_2 \cdot e_2^{i+1}$.
- Suppose that the edge from (e_1^i, e_2^i) to (e_1^{i+1}, e_2^{i+1}) is of type (2). Then $P_1 \leftarrow P_1 \cdot e_1^{i+1}$.
- Suppose that the edge from (e_1^i, e_2^i) to (e_1^{i+1}, e_2^{i+1}) is of type (3). Then $P_2 \leftarrow P_2 \cdot e_2^{i+1}$.
- Suppose that the edge from (e_1^i, e_2^i) to (e_1^{i+1}, e_2^{i+1}) is of type (4). Then $P_1 \leftarrow P_1 \cdot e_1^{i+1}$.

By the definition of edges of $\tilde{G}$, we get that P_1 and P_2 are two T-compatible vertex-disjoint paths in G such that P_i is from s_i' to t_i' for $i = 1, 2$. We can construct a graph $\tilde{G}$ in $O(|E|^3)$ time and find a path from $(s_1's_1, s_2's_2)$ to (t_1t_1', t_2t_2') in $O(|E|^3)$ time. Thus the lemma holds. □

By the results above, we can construct $\mathcal{G}$ in polynomial time. Now we show that we can solve the vertex-disjoint version of 2- DSPP WITH TRANSITION RESTRICTIONS in G by finding a path in $\mathcal{G}$ from $(s_1's_1^-, t_2't_2^+)$ to $(t_1^+t_1', s_2^-s_2')$. Note that $s_i^-, t_i^+ \in V(G')$ might be the endpoints of edges of E_0' for $i = 1, 2$. In this case, although we might contract the edges incident to $s_i^-, t_i^+ \in V(G')$ and replace these vertices with new vertices, we slightly abuse s_i^-, t_i^+ to denote the vertex adjacent to s_i', t_i' respectively in G^* for $i = 1, 2$ for the sake of simplicity.

Lemma 5.11 *There is a directed path in $\mathcal{G}$ from $(s_1's_1^-, t_2't_2^+)$ to $(t_1^+t_1', s_2^-s_2')$ if and only if G' has two vertex-disjoint T-compatible paths P_1 and P_2 such that P_i is from s_i' to t_i' and $P_i \subseteq E_i'$ for $i = 1, 2$.*

Proof ("if" direction) Suppose that G' has two vertex-disjoint T-compatible paths P_1 and P_2 such that P_i is from s_i' to t_i' and $P_i \subseteq E_i'$ for $i = 1, 2$. Recall that we contract

the edges of E'_0 in G' and reverse the edges of E^*_2 in G'' to get G^*. So by the definition of transition systems of G'' and G^*, the set $E(P_1) \setminus E'_0$ forms a directed T-compatible path P^*_1 in G^* from s'_1 to t'_1, and the set $\overline{E(P_2) \setminus E'_0}$ forms a directed T-compatible path P^*_2 in G^* from t'_2 to s'_2. Let $P^*_1 = e^0_1, e^1_1, \ldots, e^{p+1}_1$ and $e^0_1 = s'_1 s^-_1$, $e^{p+1}_1 = t^+_1 t'_1$. Let $P^*_2 = e^0_2, e^1_2, \ldots, e^{q+1}_2$ and $e^0_2 = t'_2 t^+_2$, $e^{q+1}_2 = s^-_2 s'_2$. It follows that $e^i_1 \in E^*_1$ for $i = 0, 1, \ldots, p+1$ and $e^j_2 \in E^*_2$ for $j = 0, 1, \ldots, q+1$. Since G^* is acyclic, for any $i = 0, 1, \ldots, p+1$ and for any $j = 0, 1, \ldots, q+1$, at least one of the following three cases holds.

(1) There is no directed path from $\text{head}_{G^*}(e^i_1)$ to $\text{head}_{G^*}(e^j_2)$ in G^*.

(2) There is no directed path from $\text{head}_{G^*}(e^j_2)$ to $\text{head}_{G^*}(e^i_1)$ in G^*.

(3) $\text{head}_{G^*}(e^i_1) = \text{head}_{G^*}(e^j_2)$.

By the definition of $\mathcal{G}$, the following statements hold.

- If (1) holds and $j \neq q+1$, then $\mathcal{G}$ has an edge from (e^i_1, e^j_2) to (e^i_1, e^{j+1}_2).
- If (2) holds and $i \neq p+1$, then $\mathcal{G}$ has an edge from (e^i_1, e^j_2) to (e^{i+1}_1, e^j_2).
- If (3) holds, then $\mathcal{G}$ has an edge from (e^i_1, e^j_2) to (e^{i+1}_1, e^{j+1}_2).

We can see that if $i = p+1$ then (1) holds and if $j = q+1$ then (2) holds. As a result, there is an edge from (e^i_1, e^j_2) to (e^{i+1}_1, e^j_2), (e^i_1, e^{j+1}_2) or (e^{i+1}_1, e^{j+1}_2) in $\mathcal{G}$ if $(i, j) \neq (p+1, q+1)$. It follows that starting from (e^i_1, e^j_2) with $i = 0$, $j = 0$, we can find a directed path ending at (e^{p+1}_1, e^{q+1}_2) through increasing i by 1, increasing j by 1 or increasing both i and j by 1 iteratively. This concludes the proof for "if direction".

("only if" direction) Suppose that there is a directed path from $(e^0_1, e^0_2) = (s'_1 s^-_1, t'_2 t^+_2)$ to $(e^r_1, e^r_2) = (t^+_1 t'_1, s^-_2 s'_2)$ in $\mathcal{G}$ that goes through (e^0_1, e^0_2), (e^1_1, e^1_2), …, (e^r_1, e^r_2) consecutively. We construct two T-compatible paths P_1, P_2 in G' as follows. First we initialize $P_1 = e^0_1$, $P_2 = \overline{e^0_2}$. Then for $i = 0, ..., r-1$, we update P_1 and P_2 according to the following three cases:

- Suppose that the edge from (e^i_1, e^i_2) to (e^{i+1}_1, e^{i+1}_2) is of type (i), namely $e^i_1 = e^{i+1}_1$, $\text{head}_{G^*}(e^i_2) = \text{tail}_{G^*}(e^{i+1}_2) = v$ and $\{e^i_2, e^{i+1}_2\} \in T_{G^*}(v)$. There is no path from $\text{head}_{G^*}(e^i_1)$ to v in G^*. If $v \in V'_0$, let Q be the T-compatible path in the subgraph of G' consisting of all edges of G'_v together with $f(\overline{e^i_2})$ and $f(\overline{e^{i+1}_2})$ from $\text{tail}_{G'}(f(\overline{e^{i+1}_2}))$ to $\text{head}_{G'}(f(\overline{e^i_2}))$. Then $P_2 \leftarrow f(\overline{e^{i+1}_2}) \cdot Q \setminus \{f(\overline{e^i_2}), f(\overline{e^{i+1}_2})\} \cdot P_2$. Otherwise, if $v \notin V'_0$, $P_2 \leftarrow f(\overline{e^{i+1}_2}) \cdot P_2$.
- Suppose that the edge from (e^i_1, e^i_2) to (e^{i+1}_1, e^{i+1}_2) is of type (ii), namely $e^i_2 = e^{i+1}_2$, $\text{head}_{G^*}(e^i_1) = \text{tail}_{G^*}(e^{i+1}_1) = v$ and $\{e^i_1, e^{i+1}_1\} \in T_{G^*}(v)$. There is no path from $\text{head}_{G^*}(e^i_2)$ to v in G^*. If $v \in V'_0$, let Q be the T-compatible path in the subgraph of G' consisting of all edges of G'_v together with $f(e^i_1)$ and $f(e^{i+1}_1)$ from $\text{tail}_{G'}(f(e^i_1))$ to $\text{head}_{G'}(f(e^{i+1}_1))$. Then $P_1 \leftarrow P_1 \cdot Q \setminus \{f(e^i_1), f(e^{i+1}_1)\} \cdot f(e^{i+1}_1)$. Otherwise, if $v \notin V'_0$, $P_1 \leftarrow P_1 \cdot f(e^{i+1}_1)$.
- Suppose that the edge from (e^i_1, e^i_2) to (e^{i+1}_1, e^{i+1}_2) is of type (iii), namely $\text{head}_{G^*}(e^i_1) = \text{tail}_{G^*}(e^{i+1}_1) = \text{head}_{G^*}(e^i_2) = \text{tail}_{G^*}(e^{i+1}_2) = v$ and $\{e^i_1, e^{i+1}_1\} \in$

$T_{G^*}(v), \{e_2^i, e_2^{i+1}\} \in T_{G^*}(v)$. If $v \in V_0'$, let G^s be the subgraph of G' consisting of all edges of G'_v together with $f(e_1)$, $f(e_1')$, $f(\overline{e_2})$ and $f(\overline{e_2'})$. There are two T-compatible vertex-disjoint paths in G^s, namely Q_1 from $\text{tail}_{G'}(f(e_1^i))$ to $\text{head}_{G'}(f(e_1^{i+1}))$ and Q_2 from $\text{tail}_{G'}(f(\overline{e_2^{i+1}}))$ to $\text{head}_{G'}(f(\overline{e_2^i}))$. Then $P_1 \leftarrow P_1 \cdot Q_1 \setminus \{f(e_1^i), f(e_1^{i+1})\} \cdot f(e_1^{i+1})$ and $P_2 \leftarrow f(\overline{e_2^{i+1}}) \cdot Q_2 \setminus \{f(\overline{e_2^i}), f(\overline{e_2^{i+1}})\} \cdot P_2$. Otherwise, if $v \notin V_0'$, then $P_1 \leftarrow P_1 \cdot f(e_1^{i+1})$ and $P_2 \leftarrow f(\overline{e_2^{i+1}}) \cdot P_2$.

As a result, we construct two vertex-disjoint T-compatible paths P_1 and P_2 such that P_i is from s_i' to t_i' and $P_i \subseteq E_i'$ for $i = 1, 2$. This finishes the proof for "only if" direction. □

Since $\mathcal{G}$ contains $O(|E|^3)$ edges, we can detect a path in $\mathcal{G}$ in polynomial time. Thus Lemma 5.11 shows that the vertex version of 2- DSPP WITH TRANSITION RESTRICTIONS can be solved in polynomial time assuming that every cycle in the input graph has positive length.

6 Conclusions

We initiated exploring the parameterized complexity of finding paths, cycles, and walks in forbidden-transition and edge-colored graphs. Let us contemplate a few promising directions to take the exploration further.

First, a combinatorially interesting problem eluded us during this research: The NP-hardness reduction of Szeider [49] for finding a (simple) path between two vertices of a forbidden-transition graph can be easily modified to prove that it is also NP-hard to find a (simple) compatible cycle in a forbidden-transition graph. In contrast, in edge-colored graphs finding any properly colored cycle is polynomial-time solvable [29, 52]. But what about finding a *long* properly colored cycle? More precisely, given an edge-colored graph G and an integer k we ask whether G admits a simple properly colored cycle of length at least k. Is this problem fixed-parameter tractable when parameterized by k? As the notion of properly colored walks in edge-colored graphs generalizes walks in directed graphs, the problem in question is more general than finding a cycle of length at least k in a directed graph.

For the fundamental problem of finding compatible s-t paths we obtained essentially three positive results; each leads to natural follow-up questions. First, it is not hard to obtain fixed-parameter tractability when the length of the path is explicitly bounded by the parameter or implicitly so, as in the case of the treedepth and vertex-cover parameters. We have focused here only on some salient graph parameters, in particular, graph-width parameters. It may be useful to systematically explore the hierarchy of graph parameters and classify for which parameters the problem remains tractable [50].

Second, we obtained fixed-parameter tractability parameterized by the length of the detour over the shortest s-t path. A natural question is whether one can improve on this result by considering larger lower bounds on the length of the compatible s-t path. For example, we can obtain a larger lower bound by subdividing every edge of

the input graph, replacing every vertex v by a set of length-two paths corresponding to the transitions of v, and then computing a shortest s-t path in the resulting graph.

Third, we obtained tractability for the treecut-width parameter. Other edge-cut based parameters are known [12, 19] and it would be interesting to see which of them lead to fixed-parameter algorithms, or improved running times, for finding compatible s-t paths.

Next, it is interesting to deconstruct our hardness result, to obtain potential paths to tractability [21, 38]. Our hardness result shows that finding compatible s-t paths is hard even on graphs that are close to trees in terms of vertex-deletion. However, the reduction works only if the size of the transition system and the number of vertices with forbidden transitions is not bounded by a parameter. It would hence be interesting to analyze the complexity with respect to these two parameters.

Finally, we observed in Sect. 5, that finding r disjoint shortest paths is polynomial-time solvable for $r = 2$ even if the paths are required to be compatible with a transition system. In the case of undirected graphs without forbidden transitions, this problem is polynomial-time solvable for each fixed r [8, 42]. However, the problem is open for every $r \geq 3$ in the case of directed graphs without forbidden transitions. This problem could be very interesting to generalize to forbidden-transition graphs, as it would be harder to design polynomial algorithms, but hardness would be easier to prove.

References

1. Abouelaoualim, A., Das, K.C., Faria, L., Manoussakis, Y., Martinhon, C.A.J., Saad, R.: Paths and trails in edge-colored graphs. Theor. Comput. Sci. **409**(3), 497–510 (2008)
2. Ahmed, M., Lubiw, A.: Shortest paths avoiding forbidden subpaths. In *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science (STACS 2009)*, pp. 63–74, (2009)
3. Bang-Jensen, J., Bellitto, T., Lochet, W., Yeo, A.: The directed 2-linkage problem with length constraints. Theoret. Comput. Sci. **814**, 69–73 (2020)
4. Bang-Jensen, J., Bellitto, T., Yeo, A.: Supereulerian 2-edge-coloured graphs. Technical report, arXiv:2004.01955 [math.CO], (2020)
5. Bang-Jensen, J., Gutin, G.Z.: Digraphs - Theory, Algorithms and Applications. Springer, Berlin (2009)
6. Bellitto, T.: Separating codes and traffic monitoring. *Theoretical Computer Science, Selected papers presented at the 11th International Conference on Algorithmic Aspects of Information and Management (AAIM 2016)* vol. 717, pp. 73–85 (2018)
7. Bellitto, T., Bergougnoux, B.: On minimum connecting transition sets in graphs. In Brandstädt, A., Köhler, E., Meer, K. (eds.) *Proceedings of the 44th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2018)*, volume 11159 of *Lecture Notes in Computer Science*, pp. 40–51. Springer (2018)
8. Bentert, M., Nichterlein, A., Renken, M., Zschoche, P.: Using a geometric lens to find k disjoint shortest paths. In Bansal, N., Merelli, E., Worrell, J. (eds.) *48th International Colloquium on Automata,*

Languages, and Programming (ICALP 2021), volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 26:1–26:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik
9. Bérczi, K., Kobayashi, Y.: The directed disjoint shortest paths problem. In Pruhs, K., Sohler, C. (eds.) *Proceedings of the 25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *LIPIcs*, pp. 13:1–13:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017)
10. Bezáková, I., Curticapean, R., Dell, H., Fomin, F.V.: Finding detours is fixed-parameter tractable. SIAM J. Discret. Math. **33**(4), 2326–2345 (2019)
11. Bodlaender, H.L., Cygan, M., Kratsch, S., Nederlof, J.: Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. Inf. Comput. **243**, 86–111 (2015)
12. Brand, C., Ceylan, E., Ganian, R., Hatschka, C., Korchemna, V.: Edge-cut width: An algorithmically driven analogue of treewidth based on edge cuts. In Bekos, M.A., Kaufmann, M. (eds.) *Graph-Theoretic Concepts in Computer Science*, Lecture Notes in Computer Science, pp. 98–113. Springer, Berlin (2022)
13. Contreras-Balbuena, A., Galeana-Sánchez, H., Goldfeder, I.A.: A new sufficient condition for the existence of alternating hamiltonian cycles in 2-edge-colored multigraphs. Discret. Appl. Math. **229**, 55–63 (2017)
14. Contreras-Balbuena, A., Galeana-Sánchez, H., Goldfeder, I. A.: Alternating hamiltonian cycles in 2-edge-colored multigraphs. Discrete Math. Theor. Comput. Sci., 21(1), (2019)
15. Cygan, M., Fomin, F.V., Kowalik, Ł, Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer, Berlin (2015)
16. Dorninger, D.: Hamiltonian circuits determining the order of chromosomes. Discret. Appl. Math. **50**(2), 159–168 (1994)
17. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Springer, Berlin (2013)
18. Dvorák, Z.: Two-factors in orientated graphs with forbidden transitions. Discret. Math. **309**(1), 104–112 (2009)
19. Eiben, E., Ganian, R., Hamm, T., Jaffke, L., Kwon, O.-J.: A unifying framework for characterizing and computing width measures. In *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 63:1–63:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)
20. Fellows, M.R., Hermelin, D., Rosamond, F., Vialette, S.: On the parameterized complexity of multiple-interval graph problems. Theor. Comput. Sci. **410**(1), 53–61 (2009)
21. Fellows, M.R., Jansen, B.M.P., Rosamond, F.: Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. Eur. J. Comb. **34**(3), 541–566 (2013)
22. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Berlin (2006)
23. Fortune, S., Hopcroft, J.E., Wyllie, J.: The directed subgraph homeomorphism problem. Theor. Comput. Sci. **10**, 111–121 (1980)
24. Ganian, R., Kim, E. J., Szeider, S.: Algorithmic applications of tree-cut width. In Italiano, G.F., Pighizzini, G., Sannella, D. (eds.) *Proceedings of the 40th International Symposium on Mathematical Foundations of Computer Science (MFCS 2015)*, volume 9235 of *Lecture Notes in Computer Science*, pp. 348–360. Springer, Berlin (2015)
25. Ganian, R., Ordyniak, S.: The power of cut-based parameters for computing edge disjoint paths. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pp. 190–204. Springer, Berlin (2019)
26. Gottschau, M., Kaiser, M., Waldmann, C.: The undirected two disjoint shortest paths problem. Oper. Res. Lett. **47**(1), 70–75 (2019)
27. Gourvès, L., Lyra, A., Martinhon, C.A.J., Monnot, J.: Complexity of trails, paths and circuits in arc-colored digraphs. Discret. Appl. Math. **161**(6), 819–828 (2013)
28. Gourvès, L., Lyra, A., Martinhon, C.A.J., Monnot, J., Protti, F.: On s-t paths and trails in edge-colored graphs. Electr. Notes Discrete Math. **35**, 221–226 (2009)
29. Grossman, J.W., Häggkvist, R.: Alternating cycles in edge-partitioned graphs. J. Comb. Theory Ser. B **34**(1), 77–81 (1983)
30. Gutin, G., Kim, E. J.: Properly coloured cycles and paths: results and open problems. In *Graph Theory, Computational Intelligence and Thought, Essays Dedicated to Martin Charles Golumbic on the Occasion of His 60th Birthday*, pp. 200–208, (2009)
31. Gutin, G.Z., Jones, M., Sheng, B., Wahlström, M., Yeo, A.: Chinese postman problem on edge-colored multigraphs. Discret. Appl. Math. **217**, 196–202 (2017)

32. Impagliazzo, R., Paturi, R.: Complexity of k-SAT. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity (CCC 1999)*, pp. 237–240, (1999)
33. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS 1998)*, pp. 653–662 (1998)
34. Kanté, M.M., Laforest, C., Momège, B.: Trees in graphs with conflict edges or forbidden transitions. In Chan, T.H., Lau, L.C., Trevisan, L. (eds.) *Proceedins of the 10th International Conference on Theory and Applications of Models of Computation (TAMC 2013)*, volume 7876 of *Lecture Notes in Computer Science*, pp. 343–354. Springer, Berlin (2013)
35. Kanté, M.M., Moataz, F.Z., Momège, B., Nisse, N.: Finding paths in grids with forbidden transitions. In Mayr, E.W. (ed.) *Proceedings of the 41st International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2015)*, volume 9224 of *Lecture Notes in Computer Science*, pp. 154–168. Springer, Berlin (2015)
36. Kim, E.J., Oum, S.-I., Paul, C., Sau, I., Thilikos, D.M.: An fpt 2-approximation for tree-cut decomposition. Algorithmica **80**(1), 116–135 (2018)
37. Kobayashi, Y., Sako, R.: Two disjoint shortest paths problem with non-negative edge length. Oper. Res. Lett. **47**(1), 66–69 (2019)
38. Komusiewicz, C., Niedermeier, R., Uhlmann, J.: Deconstructing intractability-a multivariate complexity analysis of interval constrained coloring. J. Discrete Algorithms **9**(1), 137–151 (2011)
39. Kotzig, A.: Moves without forbidden transitions in a graph. Matematický časopis **18**(1), 76–80 (1968)
40. Li, R., Broersma, H., Xu, C., Zhang, S.: Cycle extension in edge-colored complete graphs. Discret. Math. **340**(6), 1235–1241 (2017)
41. Li, R., Broersma, H., Zhang, S.: Properly edge-colored theta graphs in edge-colored complete graphs. Graphs Comb. **35**(1), 261–286 (2019)
42. Lochet, W.: A polynomial time algorithm for the k-disjoint shortest paths problem. In Marx, D. (ed.) *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10–13, 2021*, pp. 169–178. SIAM
43. Marx, D.: Can you beat treewidth? Theory Comput. **6**(1), 85–112 (2010)
44. Marx, D., Wollan, P.: Immersions in highly edge connected graphs. SIAM J. Discret. Math. **28**(1), 503–520 (2014)
45. Naor, M., Schulman, L.J., Srinivasan, A.: Splitters and near-optimal derandomization. In *Proceedings of 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS '95)*, pp. 182–191 (1995)
46. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press, Oxford (2006)
47. Perl, Y., Shiloach, Y.: Finding two disjoint paths between two pairs of vertices in a graph. J. ACM **25**(1), 1–9 (1978)
48. Robertson, N., Seymour, P.D.: Graph minors. XIII. The disjoint paths problem. J. Comb. Theory Ser. B **63**(1), 65–110 (1995)
49. Szeider, S.: Finding paths in graphs avoiding forbidden transitions. Discret. Appl. Math. **126**(2–3), 261–273 (2003)
50. Weller, M., Sorge, M., Contributors: The Graph Parameter Hierarchy. Accessed (October 2022)
51. Wollan, P.: The structure of graphs not admitting a fixed immersion. J. Comb. Theory Ser. B **110**, 47–66 (2015)
52. Yeo, A.: A note on alternating cycles in edge-coloured graphs. J. Comb. Theory Ser. B **69**(2), 222–225 (1997)
53. Ziobro, M., Pilipczuk, M.: Finding hamiltonian cycle in graphs of bounded treewidth: experimental evaluation. ACM J. Exp. Algorithmics **24**(1), 2.7:1-2.7:18 (2019)