Interference-free Walks in Time: Temporally Disjoint Paths

Nina Klobas[1] , **George B. Mertzios**[1*] , **Hendrik Molter**[2†] , **Rolf Niedermeier**[3] and **Philipp Zschoche**[3]

[1]Department of Computer Science, Durham University, UK

[2]Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Israel

[3]Technische Universität Berlin, Faculty IV, Algorithmics and Computational Complexity, Germany

{nina.klobas, george.mertzios}@durham.ac.uk, molterh@post.bgu.ac.il,

{rolf.niedermeier, zschoche}@tu-berlin.de

Abstract

We investigate the computational complexity of finding *temporally disjoint* paths or walks in temporal graphs. There, the edge set changes over discrete time steps and a temporal path (resp. walk) uses edges that appear at monotonically increasing time steps. Two paths (or walks) are temporally disjoint if they never use the same vertex at the same time; otherwise, they interfere. This reflects applications in robotics, traffic routing, or finding safe pathways in dynamically changing networks.

On the one extreme, we show that on general graphs the problem is computationally hard. The "walk version" is W[1]-hard when parameterized by the number of walks. However, it is polynomial-time solvable for any constant number of walks. The "path version" remains NP-hard even if we want to find only two temporally disjoint paths. On the other extreme, restricting the input temporal graph to have a path as underlying graph, quite counterintuitively, we find NP-hardness in general but also identify natural tractable cases.

1 Introduction

Computing (vertex-)disjoint paths in a graph is a cornerstone problem of algorithmic graph theory. One of the deepest achievements in discrete mathematics, graph minor theory [Robertson and Seymour, 1985; Robertson and Seymour, 1995], as well as the theory of parameterized complexity analysis [Downey and Fellows, 2013] are tightly connected to it. The problem is known to be solvable in quadratic time if the number of paths is constant, that is, it is fixed-parameter tractable when parameterized by the number of paths [Kawarabayashi *et al.*, 2012]. Besides being of fundamental interest in (algorithmic) graph theory, finding disjoint paths has many applications and there exist numerous
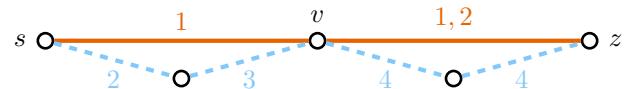

Figure 1: A temporal graph where a label of an edge reflects at which time it is available. There are two temporally disjoint (s, z)-paths P_1 and P_2, where P_1 uses the solid (orange) edges and P_2 the dashed (blue) edges. Here, P_1 uses v before P_2.

variations of the problem. In AI and robotics scenarios, for instance, multi-agent path finding is an intensively studied, closely related problem [Stern, 2019; Stern *et al.*, 2019].

Coming from the graph-algorithmic side, we propose a new view on finding disjoint paths (and walks), that is, we place the problem into the world of temporal graphs. We add a "new dimension" to the classic, static graph scenario by generalizing to a setting where the edges of a graph may appear and disappear over (discrete) time. In our model, we consider two paths (or walks) to be disjoint if they do not use the same vertex at the same point of time. Consider Fig. 1 for an example. Moreover, the path finding also has to take into account that edges are not permanently available, reflecting dynamic aspects of many real-world scenarios such as routing in traffic or communication networks, or the very dynamic nature of social networks. We intend to initiate studies on this natural scenario. Doing so, we focus on two extreme cases for the underlying graphs, namely the (underlying) graph structure being completely unrestricted or it being restricted to just a path graph. For these opposite poles, performing (parameterized) computational complexity studies, we present surprising discoveries. Before coming to these, we discuss (excerpts of) the large body of related work.

1.1 Related Work

As said, both the literature on (static) disjoint paths and its many variants as well as on multi-agent path finding is very rich. Hence, we only list a small fraction of the relevant related work. In context of graph-algorithmic work, the polynomial-time (in-)approximability of the NP-hard maximization version has been studied [Chuzhoy *et al.*, 2017]. Variants of the basic problem studied include bounds on the path length [Golovach and Thilikos, 2011] or relaxing on the disjointness of paths [Fluschnik *et al.*, 2019a;

Fluschnik *et al.*, 2019b; Guo *et al.*, 2018; Tao *et al.*, 2020].

In directed graphs, finding two disjoint paths is already NP-hard [Fortune *et al.*, 1980], whereas in directed acyclic graphs the problem is solvable in polynomial time for every fixed number of paths [Slivkins, 2010].

As to multi-agent path planning, we remark that it has been intensively researched (with several possible definitions) in the last decade in the AI and robotics communities [Almagor and Lahijanian, 2020; Atzmon *et al.*, 2020; Standley, 2010; Stern, 2019; Stern *et al.*, 2019]. Timing issues (concurrency of moving agents) or various objective functions of the agents play a fundamental role here, and also a high variety of conflict scenarios is studied.[1] The scenario we study in this work can be interpreted as a basic variant of multi-agent path planning, now translated into the world of temporal graphs.

In algorithmic graph theory, also edge-colored graphs are studied. Edge-colored graphs essentially are multilayer (or multiplex) graphs where the essential difference to temporal graphs is that there is no order on the graph layers. Here, path-finding scenarios are e.g. motivated by applications in social and optical (routing) networks [Dondi and Sikora, 2017; Santos *et al.*, 2017; Wu, 2012].

Finally, as to temporal graphs, note that several prominent graph problems have been studied in this fairly new framework. This particularly includes research on path problems [Bentert *et al.*, 2020; Enright *et al.*, 2021; Casteigts *et al.*, 2020; Mertzios *et al.*, 2019; Wu *et al.*, 2016]. In particular, another model of vertex-disjoint temporal paths [Kempe *et al.*, 2002], where two temporal paths are considered vertex-disjoint if they do not visit the same vertex. The problem of finding two such paths is NP-hard [Kempe *et al.*, 2002]. Note that the major difference to our model is that we allow two *temporally* disjoint paths to visit the same vertex as long as they do not both visit that vertex at the same *time*.

Our contributions. Our results can be grouped into two parts. First, studying temporal graphs where the underlying graph (which is obtained by making all temporal edges permanent) is unrestricted, we show that finding walks instead of paths turns out to be computationally easier. More specifically, finding temporally disjoint walks is W[1]-hard with respect to the number of walks but can be solved in polynomial time if this number is constant (i.e., we develop an XP algorithm), whereas finding temporally disjoint paths already turns out to be NP-hard for two paths. Second, restricting the input to be a temporal line (i.e., the underlying graph to be a path), we find a polynomial-time algorithm for a relevant special case while the problem remains NP-hard in general (for both paths and walks). However, we also provide a fixed-parameter tractability result with respect to the number of paths. We survey our results in Table 1. Due to space constraints, proofs of results marked with ⋆ are (partially) deferred to a full version on arXiv [Klobas *et al.*, 2021].

2 Preliminaries and Problem Definition

We denote by $\mathbb{N}$ and $\mathbb{N}_0$ the natural numbers excluding and including 0, respectively. An interval on $\mathbb{N}_0$ from a to b is denoted by $[a, b] \coloneqq \{i \in \mathbb{N}_0 \mid a \leq i \leq b\}$ and $[a] \coloneqq [1, a]$.

---

[1]Also see http://mapf.info/

Temp. Disjoint	Paths	Walks
unrestricted underlying graph	NP-hard for $\lvert S \rvert = 2$	W[1]-hard wrt. $\lvert S \rvert$ XP wrt. $\lvert S \rvert$
temporal line or tree	NP-hard FPT wrt. $\lvert S \rvert$	*open*
temporal line & S contains only pairs of extremal points	poly-time	

Table 1: Overview computational complexity of TEMPORALLY DISJOINT (PATHS/WALKS). Here, S is the multiset of source-sink pairs. Temporal line means that the underlying graph is a path.

Static graphs. An undirected graph $G = (V, E)$ consists of a set V of vertices and a set $E \subseteq \{\{v, w\} \mid v, w \in V, v \neq w\}$ of edges. For a graph G, we also denote by $V(G)$ and $E(G)$ the vertex and edge set of G, respectively. A path $P = (V, E)$ is a graph with a set $V(P) = \{v_1, \ldots, v_k\}$ of distinct vertices and edge set $E(P) = \{\{v_i, v_{i+1}\} \mid 1 \leq i < k\}$ (we often represent path P by the tuple $(v_1, v_2, \ldots, v_k)$). We say that P is a (v_1, v_k)-path and that P *visits* all vertices in $V(P)$.

A temporal graph $\mathcal{G} = (V, (E_i)_{i \in [T]})$ consists of a set V of vertices and lifetime T many edge sets $E_1, E_2, \ldots, E_T$ over V. The pair (e, i) is a *time edge* of $\mathcal{G}$ if $e \in E_i$. The graph (V, E_i) is called the i-th layer of $\mathcal{G}$. The *underlying graph* of $\mathcal{G}$ is the static graph $(V, \bigcup_{i=1}^{T} E_i)$. A *temporal (s, z)-walk* (or *temporal walk from s to z*) of length k from vertex $s = v_0$ to vertex $z = v_k$ in $\mathcal{G}$ is a sequence $P = ((v_{i-1}, v_i, t_i))_{i=1}^{k}$ of triples such that for all $i \in [k]$ we have that $\{v_{i-1}, v_i\} \in E_{t_i}$ and for all $i \in [k-1]$ we have that $t_i \leq t_{i+1}$. The *arrival time* of P is t_k. We say that P *visits* the vertices $V(P) \coloneqq \{v_i \mid i \in [0, k]\}$. In particular, P visits vertex v_i during the time interval $[t_i, t_{i+1}]$, for all $i \in [k-1]$. Furthermore, we say that P visits v_0 during time interval $[t_1, t_1]$ and P visits v_k during time interval $[t_k, t_k]$. A temporal (s, z)-walk $P = ((v_{i-1}, v_i, t_i))_{i=1}^{k}$ of length k is called a *temporal (s, z)-path* (or *temporal path from s to z*) if $v_i \neq v_j$ whenever $i \neq j$. Given two temporal walks P_1, P_2 we say that P_1 and P_2 *temporally intersect* if there exists a vertex v and two time intervals $[a_1, b_1], [a_2, b_2]$, where $[a_1, b_1] \cap [a_2, b_2] \neq \emptyset$, such that v is visited by P_1 during $[a_1, b_1]$ and by P_2 during $[a_2, b_2]$. Now, we can formally define our problem.

TEMPORALLY DISJOINT PATHS

Input: A temporal graph $\mathcal{G} = (V, (E_i)_{i \in [T]})$ and a multiset S of source-sink pairs containing elements from $V \times V$.

Question: Are there pairwise temporally non-intersecting temporal (s_i, z_i)-paths for all $(s_i, z_i) \in S$?

Analogously, TEMPORALLY DISJOINT WALKS gets the same input but asks whether there are pairwise temporally non-intersecting temporal (s_i, z_i)-walks for all $(s_i, z_i) \in S$. From the NP-hardness of DISJOINT PATHS [Karp, 1975], we immediately get the following.

Observation 1. TEMPORALLY DISJOINT (PATHS/WALKS) *is NP-hard even if $T = 1$.*

By potentially applying a linear-time preprocessing step to the input, we may assume without loss of generality that all edge sets E_i, $i = 1, 2, \ldots, T$, are non-empty. Hence, the *size* of $\mathcal{G}$ is $|\mathcal{G}| := |V| + \sum_{t=1}^{T} |E_t|$.

Parameterized complexity. Let Σ denote a finite alphabet. A parameterized problem $L \subseteq \{(x, k) \in \Sigma^* \times \mathbb{N}_0\}$ is a subset of all instances (x, k) from $\Sigma^* \times \mathbb{N}_0$, where k denotes the *parameter*. A parameterized problem L is (i) FPT (*fixed-parameter tractable*) if there is an algorithm that decides every instance (x, k) for L in $f(k) \cdot |x|^{O(1)}$ time, and (ii) contained in the class XP if there is an algorithm that decides every instance (x, k) for L in $|x|^{f(k)}$ time. where f is any computable function only depending on the parameter. If a parameterized problem L is W[1]-hard, then it is presumably not fixed-parameter tractable.

3 The Case of Few Source-Sink Pairs

In this section, we study the computational complexity of TEMPORALLY DISJOINT (PATHS/WALKS) for the case that the size of the multiset S of source-sink pairs is small. We start by showing that TEMPORALLY DISJOINT PATHS is NP-hard even for two sink-source pairs. This is a similar situation as for finding vertex-disjoint paths in directed static graphs, which is also NP-hard for two paths [Fortune *et al.*, 1980]. However, in the temporal setting there is a surprising difference between finding walks and paths that does not have an analogue in the static setting. We show that TEMPORALLY DISJOINT WALKS is W[1]-hard for the number $|S|$ of source-sink pairs and is contained in XP for the same parameter.

Theorem 2. *[⋆]* TEMPORALLY DISJOINT PATHS *is NP-hard even if $|S| = 2$ and $T = 3$.*

Proof sketch. We show that TEMPORALLY DISJOINT PATHS is NP-hard even if $|S| = 2$ and $T = 3$ by a polynomial-time reduction from the NP-complete EXACT $(3, 4)$-SAT problem [Tovey, 1984]. EXACT $(3, 4)$-SAT asks whether a Boolean formula ϕ is satisfiable, assuming that it is in conjunctive normal form, each clause has exactly three literals, and each variable appears in exactly four clauses.

Let ϕ be an instance of EXACT $(3, 4)$-SAT with n variables $x_1, x_2, \ldots, x_n$ and m clauses. We construct an instance $I = (\mathcal{G} = (V, (E_1, E_2, E_3)), S = \{(s_1, z_1), (s_2, z_2)\})$ in the following way. Intuitively, the first two layers contain the assignment gadget for the variables. The part of the temporal (s_1, z_1)-path P_1 which is in the first layer sets all variables. The temporal (s_2, z_2)-path ensures that P_1 does not "wait" in a variable gadget. In the third layer, P_1 must go from s' to z_1 through all clause gadgets. Since P_1 cannot visit a vertex twice, this validates whether the assignment satisfies ϕ. Fig. 2 depicts the resulting temporal graph.

The construction is done as follows. For each variable x_j we construct the *variable gadget* $\mathcal{G}^{x_j} = (\{a_{x_j}, a_{x_{j+1}}\} \cup \{x_j^i, \overline{x_j}^i \mid i \in [4]\}, (E_i^{x_j})_{i \in [2]})$, where $E_1^{x_j} = E_T^{x_j} \cup E_F^{x_j}$ with $E_T^{x_j} = \{\{a_{x_j}, x_j^1\}, \{x_j^4, a_{x_{j+1}}\}\} \cup \{\{x_j^i, x_j^{i+1}\} \mid i \in [3]\}$ and $E_F^{x_j} = \{\{a_{x_j}, \overline{x_j}^1\}, \{\overline{x_j}^4, a_{x_{j+1}}\}\} \cup \{\overline{x_j}^i, \overline{x_j}^{i+1} \mid$
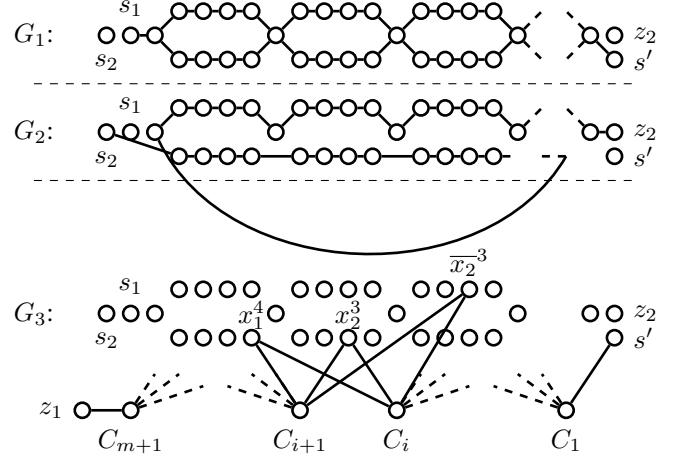

Figure 2: An excerpt of layers $G_j = (V, E_j)$, $j \in [3]$ of the temporal graph $\mathcal{G}$ used in Theorem 2. Clause $C_i = (x_1 \vee x_2 \vee \overline{x_3})$ contains the 4th appearance of x_1 and the 3rd appearance of x_2 and x_3.

$i \in [3]\}$, and $E_2^{x_j} = E_T^{x_j} \cup \{\{\overline{x_j}^i, \overline{x_j}^{i+1}\} \mid i \in [3]\} \cup \{\{\overline{x_j}^4, \overline{x_{j+1}}^1\} \mid j < n\}$. Let $C_i = (\ell_j^f \vee \ell_p^g \vee \ell_q^h)$ be a clause, where ℓ_α^β is a literal of variable x_α and its β-th appearance when iterating the clauses in the order of the indices. We now abuse our notation and say $\ell_\alpha^\beta \equiv \overline{x_\alpha}^\beta$ if ℓ_α^β is a negation of x_α and otherwise we say $\ell_\alpha^\beta \equiv x_\alpha^\beta$. We construct the *clause gadget* $\mathcal{G}^{C_i} := (\{C_i, C_{i+1}, \ell_j^f, \ell_p^g, \ell_q^h\}, (E_t^{C_i})_{t \in [3]})$ where $E_1^{C_i} = E_2^{C_i} = \emptyset$ and $E_3^{C_i} = \{\{C_r, \ell_j^f\}, \{C_r, \ell_p^g\}, \{C_r, \ell_q^h\} \mid r \in \{i, i+1\}\}$. Now we set $\mathcal{G} = (V, (E_t)_{t \in [3]})$, where $V = \bigcup_{j=1}^n V(\mathcal{G}^{x_j}) \cup \bigcup_{i=1}^m V(\mathcal{G}^{C_i}) \cup \{s_1, s_2, z_1, z_2, s'\}$, $E_1 = \bigcup_{j=1}^n E_1^{x_j} \cup \{\{s_1, a_{x_1}\}, \{a_{x_{n+1}}, s'\}\}$, $E_2 = \bigcup_{j=1}^n E_2^{x_j} \cup \{\{s_2, x_1^1\}, \{x_n^4, a_{x_1}\}, \{a_{x_{n+1}}, z_2\}\}$, and $E_3 = \bigcup_{i=1}^m E_4^{C_i} \cup \{\{s', C_1\}, \{z_1, C_{m+1}\}\}$.

Observe that I can be constructed in polynomial time. □

The reduction behind Theorem 2 heavily relies on the fact that we are dealing with paths. Indeed, for temporally disjoint walks we presumably cannot obtain NP-hardness for a constant number of sink-source pairs since, as we will show at the end of this section, TEMPORALLY DISJOINT WALKS can be solved in polynomial time if the number of source-sink pairs is constant. However, before that we show W[1]-hardness for TEMPORALLY DISJOINT WALKS parameterized by the number $|S|$ of source-sink pairs, presumably excluding the existence of an FPT-algorithm for this parameter.

Theorem 3. *[⋆]* TEMPORALLY DISJOINT WALKS *is W[1]-hard when parameterized by $|S|$, even if all edges have exactly one time label.*

We now show that TEMPORALLY DISJOINT WALKS is in XP for the parameter number $|S|$ of source-sink pairs.

Theorem 4. *[⋆]* TEMPORALLY DISJOINT WALKS *is in the class XP when parameterized by $|S|$, as it can be solved in $O(|V|^{2|S|+2} \cdot T)$ time if $|S|$ is a fixed constant.*

Proof sketch. Consider an instance $I = (\mathcal{G}, S = \{(s_1, z_1), (s_2, z_2), \ldots, (s_k, z_k)\})$ of TEMPORALLY DISJOINT WALKS. We use the following dynamic programming table D with Boolean entries. Intuitively, we want that for all $t \in \{1, \ldots, T\}$ and $v_1, \ldots, v_k \in V$ we have that $D[t, v_1, \ldots, v_k] = \top$ if and only if there are temporally non-intersecting temporal (s_i, v_i)-walks $P_1, \ldots, P_k$ with arrival time $t_i \leq t$. However, for technical reasons, we have slightly stronger requirements for D. First of all, we have a "dummy" time label zero that we use to encode the sources in the dynamic program. Formally, we initialize D as follows:

For all $v_1, \ldots v_k \in V$ we have that

$$D[0, v_1, \ldots, v_k] := \begin{cases} \top, & \text{if } \forall i \in [k]: v_i = s_i \\ \bot, & \text{otherwise.} \end{cases}$$

Furthermore, we have to model that the temporal walks we are looking for do not have to start immediately at their respective sources. Hence, if a temporal walk is still "waiting" at its source, the source vertex is not "blocked" for other temporal walks. We have a symmetric situation if temporal walks already arrived at their respective sink. In other words, if we have an entry $D[t, v_1, \ldots, v_k]$ with $v_i = v_j$ for some $i \neq j$, then it is a necessary condition for $D[t, v_1, \ldots, v_k] = \top$ that at least one of the two temporal walks i, j is either still waiting at its source or already arrived at its sink. In the latter case, we additionally need that the temporal walk arrived at the sink in a previous time step, otherwise the sink would still be blocked. We now look up in D whether there all these conditions are met for a set of temporally disjoint walks that arrive at some vertices at time $t - 1$ such that they can be extended in time step t to reach the vertices $v_1, \ldots, v_k$.

Formally, for all $t \in [T]$ we have that

$$D[t, v_1, \ldots, v_k] := \begin{cases} \top, & \begin{aligned} &\text{if } \forall i, j \in [k], v_i = v_j: \\ &\quad \exists p \in \{i, j\}: v_p \in \{s_p, z_p\}, \\ &\exists u_1, \ldots, u_k \in V: \\ &\quad (\forall i, j \in [k], i \neq j, v_i = z_i: \\ &\quad\quad u_i = z_i \vee v_i \neq v_j) \text{ and} \\ &\quad D[t-1, u_1, \ldots, u_k] = \top \text{ and} \\ &\quad ((V, E_t), \{(u_i, v_i) \mid i \in [k], \\ &\quad \{u_i, v_i\} \neq \{z_i\}, v_i \neq s_i\}) \text{ is a} \\ &\quad \textit{yes}\text{-instance of DISJ. PATHS,} \end{aligned} \\ \bot, & \text{otherwise,} \end{cases}$$

Here, DISJ. PATHS is the DISJOINT PATHS problem, where we are given an undirected graph G and set of k terminal pairs $\{(s'_i, z'_i) \mid i \in [k]\}$ and ask whether there are k vertex-disjoint paths $P_1, \ldots, P_k$ in G such that P_i is an (s'_i, z'_i)-path and P_i and P_j are vertex-disjoint for all $i \in [k]$ and $j \in [k] \setminus \{i\}$. Note that $s'_i = z'_i$ is a valid input and that in this case s'_i is the only vertex on an (s'_i, z'_i)-path. We report that I is a *yes*-instance if and only if $D[T, z_1, \ldots, z_k] = \top$. □

Finally, we point out that Theorem 2 implies that for TEMPORALLY DISJOINT PATHS we presumably cannot achieve a result similar to Theorem 4 while Theorem 3 implies that we presumably cannot improve this result on a classification level, that is we cannot expect to find an FPT-algorithm for TEMPORALLY DISJOINT WALKS for parameter $|S|$.

4 Temporal Lines and Trees

In this section, we investigate the computational complexity of TEMPORALLY DISJOINT (PATHS/WALKS) for restricted classes of underlying graphs, in particular so-called *temporal lines* and *temporal trees*. The former are temporal graphs that have a path as underlying graph and the latter are temporal graphs that have a tree as underlying graph. In particular, we first show that, surprisingly, the problems remain NP-hard on temporal lines (and thus also on temporal trees). On the positive side we show that, on temporal trees, the TEMPORALLY DISJOINT PATHS is fixed-parameter tractable with respect to the number of source-sink pairs. The latter result stands in stark contrast to the general case, where the problem is NP-hard even when the number of source-sink pairs is two (Theorem 2). If we further restrict all source-sink pairs to consist of the two end-points of the temporal line, then we obtain a polynomial-time algorithm.

Before we proceed with our results in this section, first we recall some background on foremost temporal paths. Given a temporal graph $\mathcal{G}$ and two specific vertices s, z of it, a *foremost* temporal path from s to z starting at time t is a temporal path which starts at s not earlier than at time t and arrives at z with the earliest possible arrival time. A foremost temporal path from s to z starting at time t can be computed in linear $O\left(|V| + \sum_{i=1}^{T} |E_i|\right)$ time [Wu *et al.*, 2016].

Theorem 5. *[⋆]* TEMPORALLY DISJOINT (PATHS/WALKS) *is NP-hard even on a temporal line where all temporal paths are to the same direction.*

Proof sketch. We present here a polynomial-time reduction for TEMPORALLY DISJOINT WALKS. The reduction is done from MULTICOLORED INDEPENDENT SET ON UNIT INTERVAL GRAPHS, which is known to be NP-complete [van Bevern *et al.*, 2015, Lemma 2]. In this problem the input is a unit interval graph $G = (V, E)$ with n vertices, where V is partitioned into k subsets of independent vertices; we interpret each of these subsets as a vertex color. The goal is to compute an independent set of size k in G which contains exactly one vertex from each color. By possibly slightly shifting the endpoints of the intervals in the given unit interval representation of G, we can assume without loss of generality that all endpoints of the intervals are distinct. Furthermore, we can assume without loss of generality that each interval endpoint is an integer between $k + 1$ and $k + n^2$ (while all intervals still have the same length).

From the given multi-colored unit intervals in G, we construct a temporal line $\mathcal{P}$ using the following procedure. Let $\{c_1, \ldots, c_k\}$ be a set of all colors of the intervals in G. First we fix an arbitrary linear ordering $c_1 < c_2 < \ldots < c_k$ of the k colors, and we add to the underlying path P of $\mathcal{P}$ two vertices v_i^1 and v_i^2, for every color c_i. We add to P also three basis vertices $v_\ell, v^\star, v_r$. The vertices of P are ordered starting from $v_1^1, v_2^1, \ldots, v_k^1$, followed by the basis vertices $v_\ell, v^\star, v_r$, and finishing with $v_1^2, v_2^2, \ldots, v_k^2$. At the end we have $P = (v_1^1, v_2^1, \ldots, v_k^1, v_\ell, v^\star, v_r, v_1^2, v_2^2, \ldots, v_k^2)$.

We construct the multiset S of source-sink pairs as follows. Let m_i be the number of intervals of color c_i. For every
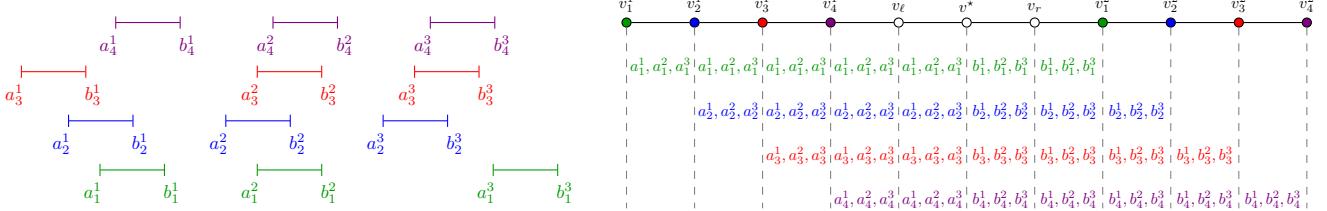
(a) Instance of the multicolored unit interval problem.

(b) Temporal graph constructed from the given multicolored unit intervals.

Figure 3: Example of the reduction described in the proof of Theorem 5.

color c_i we add the pair (v_i^1, v_i^2) to S. We refer to this source-sink pair as "the verification source-sink pair for color c_i". Furthermore, we add $m_i - 1$ copies of the pair (v_i^1, v_ℓ) to S and we add $m_i - 1$ copies of pair (v_r, v_i^2) to S. We call these $2m_i - 2$ source-sink pairs the "dummy source-sink pairs for color c_i".

To fully define the temporal line $\mathcal{P}$, we still need to add time labels to the edges of P. Denote by a_i^j and b_i^j the start and end points of the jth interval of color c_i. We set up the edge labels of the path P from v_i^1 to v_i^2 as follows. To edge $\{v_s^1, v_{s+1}^1\}$ with $s \in [k-1]$ we add the labels a_i^j with $i \leq s$. To edges $\{v_k^1, v_\ell\}$ and $\{v_\ell, v^\star\}$ we add all labels a_i^j. To edge $\{v_s^2, v_{s+1}^2\}$ with $s \in [k-1]$ we add the labels b_i^j with $i > s$. To edges $\{v^\star, v_r\}$ and $\{v_r, v_1^2\}$ we add all labels b_i^j. See Fig. 3 for an example. The construction can clearly be performed in polynomial time.

We present here the sufficiency part.

($\Leftarrow$): Assume we have a set of pairwise temporally disjoint walks for all source-sink pairs in S. Note that all edges except $\{v_\ell, v^\star\}$ and $\{v^\star, v_r\}$ have as many time labels as temporal walks that need to go through them. Furthermore, note that $\{v_\ell, v^\star\}$ has the same labels as $\{v_k^1, v_\ell\}$ and $\{v^\star, v_r\}$ has the same labels as $\{v_r, v_1^2\}$. This in particular implies that all temporal walks are in fact paths since the only vertex that could be visited by a path for more than one time step is $v^\star$. Therefore, for every pair $(s, z) \in S$, no temporal path from s to z can ever stop and wait at any vertex different from $v^\star$. Furthermore, the only paths going through vertex $v^\star$ are the paths connecting vertices v_i^1 and v_i^2 (which correspond to color c_i); we will refer to this path as the color path of c_i. Consider color c_1 and its dummy source-sink pairs (v_1^1, v_ℓ). By construction, the edge $\{v_1^1, v_2^1\}$ has time labels corresponding to the start points a_1^j of intervals from the m_1 vertices of G that have color c_1. It follows that the temporal paths for these dummy source-sink pairs and the color path of c_1 use only time labels corresponding to the start points a_1^j of intervals from the m_1 vertices of G that have color c_1 until they are at v_ℓ or arrive at $v^\star$, respectively, since they cannot wait at any vertex. Now by induction, this holds for all other colors c_i and by an analogous argument, this also holds for the "second half". More specifically, we also have that temporal paths for the dummy source-sink pairs (v_r, v_i^2) as well as the "second part" of the color path of c_i use time labels corresponding to end points b_i^j of intervals from the vertices

of G that have color c_i when going from v_r (respectively $v^\star$) to their corresponding destinations.

It follows that each color path can enter and leave vertex $v^\star$ only at the time corresponding to the start and end points of its color intervals. In any other case some of the other vertices are blocked, which prevents the completion of others temporal S-paths. Recall that intervals of the same color are non-overlapping. Hence, for every color path corresponding to a color c_i we can find one interval $[a_i^j, b_i^j]$ such that the color path visits $v^\star$ in an interval that includes $[a_i^j, b_i^j]$. Since the color paths are temporally non-intersecting, the vertices corresponding to the intervals form a multicolored independent set in G. □

Next, we show fixed-parameter tractability of TEMPORALLY DISJOINT PATHS parameterized by the number $|S|$ of source-sink pairs if the underlying graph is a tree.

Theorem 6. *[⋆] TEMPORALLY DISJOINT PATHS on temporal trees is in FPT when parameterized by $|S|$, as it can be solved in $O\left(|S|^{|S|+3} \cdot |\mathcal{G}|\right)$ time.*

Proof sketch. Let $I = (\mathcal{G}, S)$ be an instance of TEMPORALLY DISJOINT PATHS, the underlying graph G being a tree and S consists of k source-sink pairs $(s_1, z_1), \ldots, (s_k, z_k)$. We solve I using the following procedure.

First we can observe that, since G is a tree, every source-sink pair (s_i, z_i) in S corresponds to exactly one path P_i in G. Furthermore, if two paths P_1 and P_2 intersect in a tree, their intersection $P_1 \cap P_2$ is a continuous path in G (potentially containing only one vertex). In the case that P_1, P_2 intersect, there are two ways that $P_1 \cap P_2$ can be traversed: either first by P_1 and then P_2, or vice-versa. The main idea of our algorithm is that we enumerate all possible ways to traverse the intersections of each pair of paths corresponding to two source-sink pairs.

Note that the set of k paths in the tree G (where every path corresponds to one source-sink pair) have the Helly property [Bollobás, 1986]. That is, for any three of these paths P_i, P_j, P_ℓ, if they pairwise have a non-empty intersection then $P_i \cap P_j \cap P_\ell \neq \emptyset$. Therefore, in order to enumerate all possible ways to traverse the intersections of each pair of source-sink paths, it suffices to just enumerate all permutations of these k paths. □

We remark that it remains open whether a similar result can be obtained for TEMPORALLY DISJOINT WALKS, since
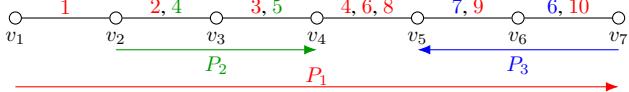
Figure 4: Temporally disjoint walks on a temporal line are not necessarily equal to temporally disjoint paths. Suppose that one wants to determine edge-labels of the following walks: P_1 from v_1 to v_7, P_2 from v_2 to v_4 and P_3 from v_7 to v_5 on the depicted temporal graph. In the feasible solution P_2 and P_3 are temporal paths, but P_1 has to be a walk.

we cannot assume w.l.o.g. that the temporally disjoint walks are actually paths, even on temporal lines (for an example see Fig. 4). Presumably (and in contrast to the general case) the walk version is computationally more difficult than the path version of our problem on temporal paths and trees.

Finally, we show that we can solve TEMPORALLY DISJOINT (PATHS/WALKS) in polynomial time if the underlying graph is a path and all source-sink pairs consist of the endpoints of that path.

Theorem 7. *[⋆] Let $\mathcal{G}$ be a temporal line having $P = (v_0, v_1, v_2, \ldots, v_n)$ as its underlying path. If S contains k times the source-sink pair (v_0, v_n) and $\ell = |S| - k$ times the source-sink pair (v_n, v_0), then* TEMPORALLY DISJOINT (PATHS/WALKS) *can be solved on $\mathcal{G}$ in polynomial time, namely $O\left(k\ell\left(k + \ell\right) \cdot |\mathcal{G}|\right)$.*

Proof sketch. We present here the proof for the problem version TEMPORALLY DISJOINT PATHS. Let $I = (\mathcal{P}, S)$ be an instance of TEMPORALLY DISJOINT PATHS, where $\mathcal{P}$ is a given temporal line with $P = (v_0, v_1, v_2, \ldots, v_n)$ as its underlying path. Assume that there have to be k (resp. $\ell = |S| - k$) temporally disjoint (v_0, v_n)- (resp. (v_n, v_0)-) paths in the output, i.e., they must have the orientation from v_0 to v_n, (resp. from v_n to v_0).

We solve the instance I using dynamic programming. The main idea is that, since all temporal paths start and end in endpoints of P, in any optimal solution, once a temporal path starts, it proceeds in the fastest possible way (without interfering with previously started paths). Therefore, assuming we start with (v_0, v_n)-temporal paths, we only need to find out *how many* (v_0, v_n)-temporal paths follow the starting path, after that *how many* (v_n, v_0)-temporal paths follow, then after that *how many* (v_0, v_n)-temporal paths follow, etc.

Let $0 \leq i \leq k$, $0 \leq j \leq \ell$, and $1 \leq t \leq T$. Then $L(i, j, t)$ denotes the earliest arrival time of $(k-i) + (\ell-j)$ temporally non-intersecting temporal paths with $k - i$ being (v_0, v_n)-temporal paths and $\ell - j$ being (v_n, v_0)-temporal paths, assuming that the earliest-starting temporal path is a (v_0, v_n)-temporal path that starts at time t. If it is not possible to route such $(k - i) + (\ell - j)$ temporally non-intersecting temporal paths starting at time t, then let $L(i, j, t) = \infty$. Similarly we define $R(i, j, t)$, with the only difference that here the earliest-starting temporal path needs to start at time t from v_n and finishes at v_0. For the sake of completeness, we let $L(i, j, \infty) = R(i, j, \infty) = \infty$ for every $i \leq k$ and every $j \leq \ell$. Furthermore, for every t, every $i \leq k - 1$, and every $j \leq \ell - 1$, we let $L(k, j, t) = R(i, \ell, t) = \infty$. Finally we let $L(k, \ell, t) = R(k, \ell, t) = t-1$. Note that, the input instance I

is a *yes*-instance if and only if $\min\{L(0, 0, 1),\ R(0, 0, 1)\} \neq \infty$. Furthermore, note that, for every triple i, j, t, the value $\min\{L(i, j, t), R(i, j, t)\}$ is the earliest arrival time of all temporal paths in the subproblem where, until time $t - 1$, exactly i and j temporally non-intersecting temporal (v_0, v_n)- and (v_n, v_0)-paths, respectively, have been routed.

The value $L(i, j, t)$ can be recursively computed as follows. Suppose that, in the optimal solution, $1 \leq p \leq k - i$ temporally non-intersecting (v_0, v_n)-temporal paths are first routed (starting at time t) before the first (v_n, v_0)-temporal path (among the $\ell - j$ ones) is routed. Let t_p be the earliest arrival time of these p paths if they can all be routed; if not, then we set $t_p = \infty$. Then:

$$L(i, j, t) = \min\{R(i + p, j, t_p + 1) \mid 1 \leq p \leq k - i\}. \quad (1)$$

The value $R(i, j, t)$ can be computed similarly:

$$R(i, j, t) = \min\{L(i, j + p, t_p^* + 1) \mid 1 \leq p \leq \ell - j\}, \quad (2)$$

where (v_n, v_0)-temporal paths are routed.

The values $\{t_p \mid 1 \leq p \leq k - i\}$ can be computed as follows. If $p = 1$ then t_p is the arrival time of the (v_0, v_n)-foremost temporal path P_1. To determine t_2, we first compute P_1 and then, for every internal vertex v_x of $\mathcal{P}$, if v_x is visited by P_1 within the time interval $[a_x, b_x]$, we remove from the edges $\{v_{x-1}, v_x\}, \{v_x, v_{x+1}\}$ of $\mathcal{P}$ all labels $l \leq b_x$. In the resulting temporal line we then compute the foremost temporal path P_2, which arrives at v_n at time t_2. By applying this procedure iteratively, we either compute p temporally non-intersecting temporal paths $P_1, P_2, \ldots, P_p$, starting at time t and arriving at time t_p, or we conclude that $t_p = \infty$. The values $\{t_p^* \mid 1 \leq p \leq \ell - j\}$ (for the (v_n, v_0)-temporal paths) can be computed in a symmetric way. All these computations together can be done in linear time.

From the above it follows that we can decide TEMPORALLY DISJOINT PATHS by checking whether $\min\{L(0, 0, 1), R(0, 0, 1)\}$ is finite or not. In total, there are $2k\ell T$ values $L(i, j, t)$ and $R(i, j, t)$. Observe that, for every pair i, j, we only need to compute the value $L(i, j, t)$ (resp. $R(i, j, t')$) for one specific value of t (resp. t'). This observation ensures that the running time of the algorithm is polynomial. $\square$

5 Conclusion

Introducing temporally disjoint paths and walks, we modeled the property that agents moving along these never meet, even though they might visit the same vertices. We identified an unexpected difference in their computational complexity: TEMPORALLY DISJOINT PATHS is NP-hard even for two paths, while TEMPORALLY DISJOINT WALKS can be done in polynomial time for a constant number of walks (however it becomes W[1]-hard when parameterized by the number of walks). On the contrary, while TEMPORALLY DISJOINT PATHS becomes fixed-parameter tractable for the number of paths if the underlying graph is a path, we leave open whether we can obtain a similar result for TEMPORALLY DISJOINT WALKS which seems to be much more complicated than the path version.

References

[Almagor and Lahijanian, 2020] S. Almagor and M. Lahijanian. Explainable multi agent path finding. In *AAMAS*, pages 34–42, 2020.

[Atzmon *et al.*, 2020] D. Atzmon, R. Stern, A. Felner, G. Wagner, R. Barták, and N.-Fa Zhou. Robust multi-agent path finding and executing. *J. Artif. Intell. Res.*, 67:549–579, 2020.

[Bentert *et al.*, 2020] M. Bentert, A.-S. Himmel, A. Nichterlein, and R. Niedermeier. Efficient computation of optimal temporal walks under waiting-time constraints. *Appl. Netw. Sci.*, 5(1):73, 2020.

[van Bevern *et al.*, 2015] R. van Bevern, M. Mnich, R. Niedermeier, and M. Weller. Interval scheduling and colorful independent sets. *J. Sched*, 18(5):449–469, 2015.

[Bollobás, 1986] B. Bollobás. *Combinatorics: Set Systems, Hypergraphs, Families of Vectors, and Combinatorial Probability*. Cambridge University Press, 1986.

[Casteigts *et al.*, 2020] A. Casteigts, A.-S. Himmel, H. Molter, and P. Zschoche. The computational complexity of finding temporal paths under waiting time constraints. In *ISAAC*, pages 30:1–30:18, 2020.

[Chuzhoy *et al.*, 2017] J. Chuzhoy, D.H.K. Kim, and R. Nimavat. New hardness results for routing on disjoint paths. In *STOC*, pages 86–99, 2017.

[Dondi and Sikora, 2017] R. Dondi and F. Sikora. Finding disjoint paths on edge-colored graphs: more tractability results. *J. Comb. Optim.*, 36:1315–1332, 2017.

[Downey and Fellows, 2013] R.G. Downey and M.R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[Enright *et al.*, 2021] Jessica Enright, Kitty Meeks, George B. Mertzios, and Viktor Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. *J. Comput. Syst. Sci.*, 119:60–77, 2021.

[Fluschnik *et al.*, 2019a] T. Fluschnik, S. Kratsch, R. Niedermeier, and M. Sorge. The parameterized complexity of the minimum shared edges problem. *J. Comput. Syst. Sci.*, 106:23–48, 2019.

[Fluschnik *et al.*, 2019b] T. Fluschnik, M. Morik, and M. Sorge. The complexity of routing with collision avoidance. *J. Comput. Syst. Sci.*, 102:69–86, 2019.

[Fortune *et al.*, 1980] S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theor. Comput. Sci.*, 10(2):111–121, 1980.

[Golovach and Thilikos, 2011] P.A. Golovach and D.M. Thilikos. Paths of bounded length and their cuts: parameterized complexity and algorithms. *Discrete Optim.*, 8(1):72–86, 2011.

[Guo *et al.*, 2018] L. Guo, Y. Deng, K. Liao, Q. He, T. Sellis, and Z. Hu. A fast algorithm for optimally finding partially disjoint shortest paths. In *IJCAI*, pages 1456–1462, 2018.

[Karp, 1975] R.M. Karp. On the computational complexity of combinatorial problems. *Networks*, 5(1):45–68, 1975.

[Kawarabayashi *et al.*, 2012] K. Kawarabayashi, Y. Kobayashi, and B. Reed. The disjoint paths problem in quadratic time. *J. Comb. Theory, Ser. B*, 102(2):424–435, 2012.

[Kempe *et al.*, 2002] D. Kempe, J. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. *J. Comput. Syst. Sci.*, 64(4):820–842, 2002.

[Klobas *et al.*, 2021] Nina Klobas, George B. Mertzios, Hendrik Molter, Rolf Niedermeier, and Philipp Zschoche. Interference-free walks in time: Temporally disjoint paths. *CoRR*, abs/2105.08335, 2021.

[Mertzios *et al.*, 2019] G.B. Mertzios, O. Michail, and P.G. Spirakis. Temporal network optimization subject to connectivity constraints. *Algorithmica*, 81(4):1416–1449, 2019.

[Robertson and Seymour, 1985] N. Robertson and P.D. Seymour. Disjoint paths—a survey. *SIAM J. Algebraic Discrete Methods*, 6(2):300–305, 1985.

[Robertson and Seymour, 1995] N. Robertson and P.D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.

[Santos *et al.*, 2017] R.F. Santos, A. Andrioni, A.C. Drummond, and E.C. Xavier. Multicolour paths in graphs: NP-hardness, algorithms, and applications on routing in WDM networks. *J. Comb. Optim.*, 33(2):742–778, 2017.

[Slivkins, 2010] A. Slivkins. Parameterized tractability of edge-disjoint paths on directed acyclic graphs. *SIAM J. Discrete Math.*, 24(1):146–157, 2010.

[Standley, 2010] T.S. Standley. Finding optimal solutions to cooperative pathfinding problems. In *AAAI*, pages 173–178, 2010.

[Stern *et al.*, 2019] R. Stern, N.R. Sturtevant, A. Felner, S. Koenig, H. Ma, T.T. Walker, J. Li, D. Atzmon, L. Cohen, T.K.S. Kumar, R. Barták, and E. Boyarski. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *SOCS*, pages 151–159, 2019.

[Stern, 2019] R. Stern. Multi-agent path finding - an overview. In *Artificial Intelligence - 5th RAAI Summer School*, pages 96–115. Springer, 2019.

[Tao *et al.*, 2020] B. Tao, M. Xiao, and J. Zhao. Finding minimum-weight link-disjoint paths with a few common nodes. In *AAAI*, pages 938–945, 2020.

[Tovey, 1984] C.A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Appl. Math.*, 8(1):85–89, 1984.

[Wu *et al.*, 2016] H. Wu, J. Cheng, Y. Ke, S. Huang, Y. Huang, and H. Wu. Efficient algorithms for temporal path computation. *IEEE Trans. Knowl. Data Eng.*, 28(11):2927–2942, 2016.

[Wu, 2012] B.Y. Wu. On the maximum disjoint paths problem on edge-colored graphs. *Discrete Optim.*, 9(1):50–57, 2012.