In Which Graph Structures Can We Efficiently Find Temporally Disjoint Paths and Walks?

Pascal Kunz[1,2] , **Hendrik Molter**[3] and **Meirav Zehavi**[3]
[1]Humboldt-Universität zu Berlin, Algorithm Engineering, Berlin, Germany
[2]Technische Universität Berlin, Algorithmics and Computational Complexity, Berlin, Germany
[3]Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel
p.kunz.1@tu-berlin.de, molterh@post.bgu.ac.il, meiravze@bgu.ac.il

Abstract

A temporal graph has an edge set that may change over discrete time steps, and a temporal path (or walk) must traverse edges that appear at increasing time steps. Accordingly, two temporal paths (or walks) are temporally disjoint if they do not visit any vertex at the same time. The study of the computational complexity of finding *temporally disjoint* paths or walks in temporal graphs has recently been initiated by Klobas *et al.*. This problem is motivated by applications in multi-agent path finding (MAPF), which include robotics, warehouse management, aircraft management, and traffic routing.

We extend Klobas *et al.*'s research by providing parameterized hardness results for very restricted cases, with a focus on structural parameters of the so-called underlying graph. On the positive side, we identify sufficiently simple cases where we can solve the problem efficiently. Our results reveal some surprising differences between the "path version" and the "walk version" (where vertices may be visited multiple times) of the problem, and answer several open questions posed by Klobas *et al.*

1 Introduction

Deciding whether a set of vertex pairs (called source-sink pairs) in a graph can be connected by pairwise vertex disjoint paths is a problem that is of fundamental interest in algorithmic graph theory. It was among the first problems that were shown to be NP-complete [Karp, 1975] and the further study of the problem is closely tied to one of the most ground-breaking achievements in discrete mathematics in recent history, graph minor theory [Neil and Seymour, 1985; Neil and Seymour, 1995]. The disjoint path problem is known to be solvable in quadratic time if the number of vertex pairs that need to be connected is constant, that is, the problem is fixed-parameter tractable for the number of sought paths [Kawarabayashi *et al.*, 2012]. On directed graphs, finding two disjoint paths is NP-hard [Fortune *et al.*, 1980], but on directed acyclic graphs the problem is solvable in polynomial time if the number of paths is a constant [Slivkins, 2010].

Klobas *et al.* [2023] recently introduced and studied two natural *temporal* versions of the disjoint path problem, called TEMPORALLY DISJOINT PATHS (TDP) and TEMPORALLY DISJOINT WALKS (TDW). Informally speaking, a temporal graph has an edge set that may change over discrete time steps. Accordingly, temporal paths must traverse edges that appear at increasing time steps and may visit each vertex at most once, whereas a temporal walk may visit each vertex multiple times. Furthermore, two temporal paths (or walks) are *temporally disjoint* if they do not occupy any vertex at the same time. Analogously to the non-temporal setting, the goal is to find temporal paths (or walks) connecting vertex pairs of a given multiset such that those paths (or walks) are pairwise temporally disjoint. We give formal definitions in Section 2.

Due to the asymmetric and non-transitive nature of connectivity in temporal graphs, path-finding related problems behave quite differently in the temporal setting than in the static setting. In fact, there are many natural temporal path-finding problems that do not have a direct analog in the static setting [Casteigts *et al.*, 2021; Füchsle *et al.*, 2022]. For TDP and TDW the situation is similar. Among other results, Klobas *et al.* [2023], for example, showed that both problems are NP-hard if the underlying graph[1] is a path, a setting where the static disjoint path problem is trivial. Furthermore, they revealed surprising differences in the computational complexity of TDP and TDW. We build on the work of Klobas *et al.* [2023] and continue the study of the (parameterized) computational complexity of TDP and TDW. We provide several new hardness and algorithmic results that expose further interesting differences of the two problem variants and that resolve some of the open questions by Klobas *et al.* [2023].

One of the main application areas for the temporal disjoint path problems is multi-agent path finding (MAPF), an area that has attracted a lot of research from the AI and robotics community in recent years [Stern, 2019; Stern *et al.*, 2019; Salzman and Stern, 2020]. The goal is to find paths for multiple agents such that all agents can follow these paths concurrently without colliding. The main difference between classical disjoint path problems and the basic setting of multi-agent path finding problems is that in the latter, we assume the agents move along the paths one step at a time and only collide when they move to the same vertex at the same time. This means that the paths in a solution to a MAPF problem

[1]The *underlying graph* of a temporal graph is the static graph containing all edges that appear at least once in the temporal graph.

are not necessarily vertex disjoint, but if two paths have a common vertex, that vertex cannot be at the same ordinal position in both paths. A key difference between classical MAPF settings and the problems we study is the assumption that the network structure may change over time. Applications of MAPF include autonomous vehicles, robotics, automated warehouses, and airport towing [Stern, 2019; Stern *et al.*, 2019; Salzman and Stern, 2020], where predictable changes over time in the network topology are well-motivated in many real-world scenarios [Latapy *et al.*, 2018; Michail, 2016; Holme and Saramäki, 2019].

Related Work. The (non-temporal) disjoint path problem is one of the most central problems in algorithmic graph theory, and hence has been extensively studied in the literature for the past few decades. For an overview, we refer to Korte *et al.* [1990].

In recent years there has also been intensive research on (non-temporal) multi-agent path finding problems (MAPF) in multiple variations, mostly in the AI and robotics community [Stern, 2019; Stern *et al.*, 2019; Salzman and Stern, 2020]. MAPF can hence be seen as a generalization of so-called *pebble motion problems* on graphs and it is known to be NP-hard [Goldreich, 2011; Yu and LaValle, 2013]. To the best of our knowledge, the current state of the art (optimal) algorithms for MAPF problems employ the so-called conflict-based search approach [Sharon *et al.*, 2015].

In MAPF, various settings and problem variations have been considered, including cooperative settings [Standley, 2010], robustness requirements [Atzmon *et al.*, 2020] or presence of delays [Ma *et al.*, 2017], online settings [Švancara *et al.*, 2019], continuous time settings [Andreychuk *et al.*, 2022], explainability requirements [Almagor and Lahijanian, 2020], settings with large agents (that occupy multiple vertices) [Li *et al.*, 2019], and many more. For an extensive overview, we refer to Stern [2019] and Stern *et al.* [2019].

Klobas *et al.* [2023] started the study of TDP and TDW, which can be interpreted as MAPF settings where the availability of edges in the graph may change while the agents are moving. Using the MAPF terminology of Stern *et al.* [2019], the problem setting considers both so-called "vertex-conflicts" and "edge-conflicts" between agents, that is, two agents cannot occupy the same vertex at the same time and cannot traverse the same edge at the same time. Furthermore, it uses the "disappear at target" assumption, that is, once an agent reaches their target vertex, another agent can occupy this vertex again. It also uses an "appear at start" assumption, that is, an agent is deployed to their starting vertex when they start moving, and other agents can occupy the starting vertex before. We remark that the latter two assumptions are not crucial for many of our results. We discuss this issue in more detail in the full version of this work [Kunz *et al.*, 2023]. We point out that Klobas *et al.* [2023] consider so-called *non-strict* temporal paths, that traverse edges that appear at non-decreasing time steps. In this work, we consider so-called *strict* temporal paths that, as described earlier, must traverse edges that appear at (strictly) increasing time steps. The latter models the common assumption in MAPF, that agents can move along at most one edge at each time step [Stern, 2019]. A closer inspection of the proofs by Klobas *et al.* [2023] reveals that the results we mention in the following also hold for the strict case. TDP is NP-hard even for two source-sink pairs, whereas TDW is W[1]-hard for the number of source-sink pairs but can be solved in polynomial time for a constant number of source-sink pairs [Klobas *et al.*, 2023]. Furthermore, both problem variants are NP-hard even if the underlying graph is a path [Klobas *et al.*, 2023]. Nevertheless, TDP is fixed-parameter tractable with respect to the number of source-sink pairs if the underlying graph is a forest [Klobas *et al.*, 2023].

Finally, we remark that a different version of disjoint paths in temporal graphs has been studied by Kempe *et al.* [2002]. They consider two temporal paths to be disjoint if they to not visit a common vertex, even if that vertex is not occupied by the two temporal paths at the same time. They show that finding two such paths in NP-hard.

Our Contribution. The goal of our work is to further understand which structures of the underlying graph can be exploited to solve TDP and TDW efficiently (in terms of parameterized computational complexity). Due to space constraints, proofs of statements marked with $\star$ are (partially) deferred to the full version [Kunz *et al.*, 2023]. Our first main computational hardness result shows that presumably, we cannot solve TDP and TDW efficiently in the very restricted case where the number of vertices is small and the underlying graph is a star (a center vertex connected to leaves).

- TDP and TDW are NP-hard and W[1]-hard for the number of vertices even if the underlying graph is a star.

Recall that we have a multiset of source-sink pairs, that is, the number of source-sink pairs in the input may be much larger than the number of vertices. This leads us to focusing on cases where we consider the number of source-sink pairs as (part of) the parameter. As mentioned before, Klobas *et al.* [2023] showed that TDP is fixed-parameter tractable with respect to the number of source-sink pairs if the underlying graph is a forest. They left open whether this algorithm can be generalized to an FPT-algorithm where the parameter is the number of source-sink pairs combined with some distance-to-forest measure for the underlying graph. They also left open whether a similar algorithm can be found for TDW. We resolve both of these open questions.

For TDP we show that we presumably cannot obtain an FPT-algorithm even if we combine the number of source-sink pairs with the vertex cover number of the underlying graph.

- TDP is W[1]-hard for the combination of the number of source-sink pairs and the vertex cover number of the underlying graph.

This result excludes several popular distance-to-forest measures as potential parameters, such as the treewidth or the feedback vertex number, which are smaller than the vertex cover number. On the positive side, we can show that we can use the feedback edge number (which is a distance-to-forest measure that is incomparable to the vertex cover number) as an additional parameter to obtain tractability.

- TDP is in FPT with respect to the combination of the number of source-sink pairs and the feedback edge number of the underlying graph.

The parameterized complexity of TDW is surprisingly different. For this problem we can show that, in contrast to the path version, we presumably cannot obtain an FPT-algorithm for the number of source-sink pairs even if the underlying graph is a star.

- TDW is W[1]-hard for the number of source-sink pairs even if the underlying graph is a star.

This is quite surprising given that Klobas *et al.* [2023] showed that TDW is easier to solve than TDP when the number of source-sink pairs is considered as a parameter and the underlying graphs is unrestricted. As mentioned before, they showed that in general, TDW can be solved in polynomial time if the number of source-sink pairs is constant whereas TDP is NP-hard already for two source-sink pairs. On the positive side, if the underlying graph is restricted to be a path, we can achieve fixed-parameter tractability for TDW.

- TDW is in FPT with respect to the number of source-sink pairs if the underlying graph is a path.

Our results provide a quite complete picture of the parameterized complexity of TDP and TDW when the number of source-sink pairs and structural parameters (particularly ones that measure similarity to forests) of the underlying graph are considered. We point out remaining open cases and future research directions in Section 5.

2 Preliminaries and Problem Definitions

An interval of non-negative integers from a to b is denoted by $[a, b] := \{i \in \mathbb{N} \cup \{0\} \mid a \leq i \leq b\}$ and $[a] := [1, a]$. An (s, z)-*walk* (or *walk* from s to z) in a graph $G = (V, E)$ of length k from vertex $s = v_0$ to vertex $z = v_k$ is a sequence $P = (v_{i-1}, v_i)_{i=1}^{k}$ of *static transitions* such that for all $i \in [k]$ we have that $\{v_{i-1}, v_i\} \in E$. The (s, z)-walk P is called an (s, z)-*path* (or *path* from s to z) if $v_i \neq v_j$ whenever $i \neq j$.

A *temporal graph* $\mathcal{G} = (V, E_1, \ldots, E_T)$ or $\mathcal{G} = (V, (E_t)_{t\in[T]})$ consists of a vertex set V and T edge sets $E_1, \ldots, E_T \subseteq \binom{V}{2}$. The *underlying graph* of $\mathcal{G}$ is the static graph $G_U = (V, E_U)$ with $E_U := \bigcup_{i=1}^{T} E_i$. The indices $1, \ldots, T$ are the *time steps* of $\mathcal{G}$. The temporal graph $\mathcal{G}$ is a *temporal tree*, *star*, or *line*, respectively, if its underlying graph is a tree, star, or path. We call the pair (e, i) a *time edge* of $\mathcal{G}$ if $e \in E_i$. The graph (V, E_i) is the *i-th layer* of $\mathcal{G}$.

A *temporal* (s, z)-*walk* (or *temporal walk* from s to z) in $\mathcal{G}$ of length k from vertex $s = v_0$ to vertex $z = v_k$ is a sequence $P = ((v_{i-1}, v_i, t_i))_{i=1}^{k}$ of *transitions* 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 $t_i < t_{i+1}$. The temporal (s, z)-walk P is called a *temporal* (s, z)-*path* (or *temporal path* from s to z) if $v_i \neq v_j$ whenever $i \neq j$. The *arrival time* of P is t_k. We say that P *visits* the vertices $V(P) := \{v_i \mid i \in [0, k]\}$ in order $v_0, v_1, \ldots, v_k$. We say that P *occupies* vertex v_i during the time interval $[t_i, t_{i+1}]$, for all $i \in [k-1]$. Furthermore, we say that P occupies v_0 during time interval $[t_1, t_1]$ and P occupies v_k during time interval $[t_k, t_k]$. We say that P *follows* the (static) path or walk $P' = (v_{i-1}, v_i)_{i=1}^{k}$ of the underlying graph of $\mathcal{G}$. If the arrival time of P is the smallest possible among all temporal (s, z)-walks in $\mathcal{G}$, we call P *foremost*. If for all v_i with $i \in [k]$ we have that t_i is the arrival time of a foremost (s, v_i)-path in $\mathcal{G}$, then we call P *prefix foremost*. If there is a temporal (s, z)-path in $\mathcal{G}$, then there also exists a prefix-foremost (s, z)-path and such a path can be computed in polynomial time [Wu *et al.*, 2016]. 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 occupied by P_1 during $[a_1, b_1]$ and by P_2 during $[a_2, b_2]$. We say that P_1 and P_2 are *temporally disjoint* if they are not temporally intersecting. The problem TDP is formally defined as follows.

TEMPORALLY DISJOINT PATHS (TDP)	
Input:	A temporal graph $\mathcal{G} = (V, (E_t)_{t\in[T]})$ and a multiset S of source-sink pairs containing elements from $V \times V$.
Question:	Are there pairwise temporally disjoint temporal (s_i, z_i)-paths for all $(s_i, z_i) \in S$?

The problem TEMPORALLY DISJOINT WALKS (TDW) receives the same input but asks whether there are pairwise temporally disjoint temporal (s_i, z_i)-walks for all $(s_i, z_i) \in S$. Given an instance of TDP or TDW, we use $\hat{S}$ to denote the set of vertices in V that appear as sources or sinks in S, that is, $\hat{S} = \{s \in V \mid (s, z) \in S \text{ for some } z \in V\} \cup \{z \in V \mid (s, z) \in S \text{ for some } s \in V\}$.

We study the (parameterized) computational complexity of those two problems. We use the following standard concepts from parameterized complexity theory [Downey and Fellows, 2013; Flum and Grohe, 2006; Cygan *et al.*, 2015]. A *parameterized problem* $L \subseteq \Sigma^* \times \mathbb{N}$ is a subset of all instances (x, k) from $\Sigma^* \times \mathbb{N}$, where k denotes the *parameter*. A parameterized problem L is in the class FPT (or *fixed-parameter tractable*) if there is an algorithm that decides every instance (x, k) for L in $f(k) \cdot |x|^{O(1)}$ time, where f is any computable function that depends only on the parameter. If a parameterized problem L is W[1]-hard, then it is presumably not fixed-parameter tractable [Downey and Fellows, 2013; Flum and Grohe, 2006; Cygan *et al.*, 2015].

3 Parameterized Hardness of TDP and TDW

In this section, we analyze the parameterized hardness of TDP and TDW with respect to structural parameters of the underlying graph (combined with the number of source-sink pairs). We first consider the number $|V|$ of vertices in the input temporal graph as a parameter, which one might consider as the largest structural parameter of the underlying graph.

Theorem 1 ($\star$). *TDP and TDW on temporal stars are NP-hard and W[1]-hard with respect to the number of vertices.*

Theorem 1 is proved by a parameterized reduction from the W[1]-hard problem UNARY BIN PACKING parameterized by the number of bins [Jansen *et al.*, 2013]. This reduction is deferred to the full version [Kunz *et al.*, 2023].

3.1 Parameterized Hardness of TDP

In this section, we show that TDP is W[1]-hard when parameterized by the combination of source-sink pairs and the vertex cover number of the underlying graph. This shows to which extend we can expect to generalize the FPT-algorithm for TDP for the number of source-sink pairs on temporal forests by [Klobas *et al.*, 2023]. Our result rules out FPT-algorithms for TDP parameterized by the number of source-sink pairs combined with e.g. the treewidth or the feedback vertex number of the underlying graph, since both of these parameters are smaller than the vertex cover number. To obtain tractability, we have to use parameters that are larger or incomparable to the vertex cover number, such as the feedback edge number. In Section 4.1, we show this is possible.

Theorem 2 ($\star$)**.** TDP *is W[1]-hard when parameterized by the combination of the number $|S|$ of source-sink pairs and the vertex cover number of the underlying graph.*

Theorem 2 is proved by a parameterized reduction from the W[1]-hard problem MULTICOLORED CLIQUE parameterized by the number of colors [Fellows *et al.*, 2009]. The reduction is deferred to the full version [Kunz *et al.*, 2023].

3.2 Parameterized Hardness of TDW

Klobas *et al.* [2023] left the parameterized complexity of TDW with respect to the number $|S|$ of source-sink pairs on temporal trees as an open question. In this section, we answer this question by showing that the problem is W[1]-hard for this parameterization, even on temporal stars. This may be somewhat surprising considering that Klobas *et al.* [2023] showed that TDP is fixed-parameter tractable on trees. This implies that while TDP is harder than TDW on arbitrary graphs for $|S|$ as the parameter (the former is NP-hard for $|S| = 2$, while the latter is solvable in polynomial time for constant $|S|$), TDW is harder than TDP on temporal trees.

Theorem 3 ($\star$)**.** TDW *on temporal stars is* W[1]*-hard when parameterized by the number $|S|$ of source-sink pairs.*

We will give a parameterized reduction from the W[1]-hard [Fellows *et al.*, 2009] problem MULTICOLORED CLIQUE. The input for this problem consists of an integer k and a properly k-colored graph $G = (V_1 \uplus V_2 \uplus \ldots \uplus V_k, E)$ and one is asked to decide whether G contains a clique of size k. Any such clique must, of course, contain exactly one vertex from each color class.

Construction 1. Let $(G = (V_1 \uplus V_2 \uplus \ldots \uplus V_k, E), k)$ be an instance of MULTICOLORED CLIQUE. We may assume w.l.o.g. that $|V_1| = |V_2| = \ldots |V_k| =: n$. Suppose that $V_i = \{v_1^i, \ldots, v_n^i\}$. We will now construct an instance $(\mathcal{G} = (V, (E_t)_{t\in[T]}), S)$ of TDW.

We start by describing S. For every $i \in [k]$, there are two terminal pairs (s_i, z_i) and $(\tilde{s}_i, \tilde{z}_i)$. The temporal walks that connect these two pairs will encode the selection of a vertex in V_i. Additionally, for every $i, j \in [k]$ with $i < j$ there is a terminal pair $(s_{i,j}, z_{i,j})$. The temporal walk for this pair verifies that at least one vertex has been selected in each of V_i and V_j and that those two vertices are adjacent. Let S denote this set of terminal pairs.

Next we will define $\mathcal{G} = (V, E_1, \ldots, E_T)$. We start by giving V. For every $i \in [k]$ there are two sets of vertices, the first, W_i, is intended to be used by the temporal walk connecting (s_i, z_i) and the other, $\tilde{W}_i$, by the temporal walk for $(\tilde{s}_i, \tilde{z}_i)$. Let $W_i := \{w_1^i, \ldots, w_{kn}^i, x_1^i, \ldots, x_{kn}^i, y_1^i, \ldots, y_n^i\}$ and $\tilde{W}_i := \{\tilde{w}_1^i, \ldots, \tilde{w}_{kn}^i, \tilde{x}_1^i, \ldots, \tilde{x}_{kn}^i, \tilde{y}_1^i, \ldots, \tilde{y}_n^i\}$. Then, for every edge $e = \{u, v\} \in E$ with $u \in V_i$ and $v \in V_j$ for some $i < j$ there are vertices $W_e := \{\alpha_u^j, \beta_e, \gamma_v^i\}$. Finally, there is a central vertex c, to which every edge will be incident. Let $V := \{c\} \cup \left(\bigcup_{(s,z)\in S}\{s, z\}\right) \cup \left(\bigcup_{i\in[k]} W_i \cup \tilde{W}_i\right) \cup \left(\bigcup_{e\in E} W_e\right)$.

It remains to define $E_1, \ldots, E_T$. For every $i \in [k]$, there is a sequence of edge sets $E_0^i, \ldots, E_{4kn+6}^i$. Informally speaking, in this sequence the temporal walks connecting (s_i, z_i) and $(\tilde{s}_i, \tilde{z}_i)$ select a vertex in V_i. The temporal walk connecting $(s_{i,j}, z_{i,j})$ for $j \neq i$ verifies that the selected vertex is adjacent to the one selected in V_j. First, for $\ell \in [kn]$, the vertices w_ℓ^i are adjacent to c in the layers $E_{4\ell-3}^i$ and $E_{4\ell}^i$, and the vertices $\tilde{w}_\ell^i$ have an edge to c in $E_{4\ell-1}^i$ and $E_{4\ell+2}^i$. Next, for $\ell \in [kn]$, the vertices x_ℓ^i are adjacent to c in $E_{4\ell-2}^i$ and $E_{4\ell+1}^i$ and $\tilde{x}_\ell^i$ have edges to c in $E_{4\ell}^i$ and $E_{4\ell+3}^i$. Finally, for $\ell \in [n]$, the vertices y_ℓ^i are adjacent to c in the layers $E_{4k(\ell-1)+1}^i$ and $E_{4k\ell+1}^i$, while for $\tilde{y}_\ell^i$ those layers are $E_{4k(\ell-1)+3}^i$ and $E_{4k\ell+3}^i$. Additionally, the starting vertex s_i is adjacent to c in the layer E_0^i and z_i has an edge to c in E_{4kn+4}^i. Similarly, for $\tilde{s}_i$ and $\tilde{z}_i$, those layers are E_2^i and E_{4kn+6}^i, respectively.

For any $i, j \in [k]$ with $i < j$, there is a layer $E_1^{i,j}$, which contains the edge $\{s_{i,j}, c\}$ and a second subsequent layer $E_2^{i,j}$, which contains $\{c, \alpha_v^j\}$ for all $v \in V_i$ that have a neighbor in V_j. There is also a layer $E_{f-1}^{i,j}$, which connects c to γ_v^i for all $v \in V_j$ that have a neighbor in V_i, and finally $E_f^{i,j}$ connecting $z_{i,j}$ to c. Next, consider edge $e \in E$. Suppose that one endpoint of that edge is $v_a^i \in V_i$ and the other endpoint is $v_b^j \in V_j$, with $i < j$. Then, there is an edge from $\alpha_{v_a^i}^j$ to c in the layer $E_{4k(a-1)+4(j-1)}^i$, from c to β_e in layer $E_{4k(a-1)+4(j-1)+2}^i$, from β_e to c in $E_{4k(b-1)+4i}^j$, and from c to $\gamma_{v_b^j}^i$ in $E_{4k(b-1)+4i+2}^j$.

The order of the layers in $\mathcal{G}$ is as follows. The layers $E_1^{i,j}$ and $E_2^{i,j}$ for each $i, j \in k$ are consecutive to one another and all such layers come at the very beginning of the temporal graph. Then, come the layers $E_1^1, \ldots, E_{4kn+6}^1$, followed by $E_1^2, \ldots, E_{4kn+6}^2$, and so on. The temporal graph concludes with the layers $E_{f-1}^{i,j}$ and $E_f^{i,j}$ consecutively for each $i, j \in k$. We give an illustration of the construction in Figure 1. $\diamond$

We will now give a brief overview of the intuition as to why this construction is correct before proving this claim formally. First, consider for any $i \in [k]$ temporal walks P_i and $\tilde{P}_i$ that connect (s_i, z_i) and $(\tilde{s}_i, \tilde{z}_i)$, respectively. For the purpose of explanation, assume for now that the vertices y_ℓ^i and $\tilde{y}_\ell^i$ did not exist. The first walk, P_i, must move from s_i to c in layer E_0^i, because s_i is subsequently isolated. The second walk, $\tilde{P}_i$, must similarly arrive in c in E_2^i. Hence, P_i must leave c in E_1^i, otherwise it temporally intersects $\tilde{P}_i$. It must move to w_1^i. That vertex is again adjacent to c in layer E_4^i and isolated after that. Hence, P_i must return to c in that layer. Therefore, $\tilde{P}_i$ must leave c in layer E_3^i, otherwise it temporally intersects P_i. It must move to $\tilde{w}_1^i$. We can see that the two walks $P_i, \tilde{P}_i$ are "locked" into alternatingly moving from the center c to vertices w_ℓ^i and $\tilde{w}_\ell^i$, respectively. For an illustration see Figure 1.

Now consider that the vertices y_ℓ^i and $\tilde{y}_\ell^i$ exist. The pattern in which P_i and $\tilde{P}_i$ move can be "broken" if P_i or $\tilde{P}_i$ moves to y_ℓ^i or $\tilde{y}_\ell^i$, respectively. Now we can make two observations.

- If the two walks do so almost simultaneously it creates an interval of size $O(k)$ where neither of the two walks occupy the center vertex c. This interval corresponds to a vertex in V_i. Hence, by choosing when to move to y_ℓ^i and $\tilde{y}_\ell^i$, respectively, a vertex from V_i is "selected".
- After the two walks move back to c, they are locked in a similar pattern, where they alternatingly move from the center to vertices x_ℓ^i and $\tilde{y}_\ell^i$, respectively. (This happens also if only one of the walks move to y_ℓ^i or $\tilde{y}_\ell^i$.) From this pattern, they cannot move to vertices y_ℓ^i or $\tilde{y}_\ell^i$, hence at most one vertex is selected per color.

If no vertex is selected, that is, no vertices y_ℓ^i or $\tilde{y}_\ell^i$ are visited, then the temporal walk $P_{i,j}$ from $s_{i,j}$ to $z_{i,j}$ for any $i < j$ will temporally intersect P_i or $\tilde{P}_i$. (If $i = k$ we make an analogous observation, where i and j exchange their role.) Hence, we can assume that one vertex of every color is selected. We can observe that $P_{i,j}$ first must move from $s_{i,j}$ to c and then to α_e where e is any edge connecting a vertex in V_i to a vertex in V_j. Informally speaking, this is only possible without temporally intersecting any of the temporal walks $P_i, \tilde{P}_i, P_j, \tilde{P}_j$ if the endpoints of e are selected as vertices for colors i and j, respectively. Thereby, we verify that the selected vertices indeed form a clique in G.

The formal correctness proof for our reduction is deferred to the full version [Kunz *et al.*, 2023].

4 Algorithms for TDP and TDW

In this section, we present two new algorithms, one for TDP and one for TDW.

4.1 Algorithm for TDP

In this section, we present an FPT-algorithm for TDP parameterized by the combination of the number of source-sink pairs and the feedback edge number[2] of the underlying graph. This generalizes the FPT-algorithm by Klobas *et al.* [2023] for TDP parameterized by the number of source-sink pairs for temporal forests. Theorem 2 implies that we presumably cannot replace the feedback edge number of the underlying graph by a smaller parameter such as feedback vertex number or treewidth and still obtain fixed-parameter tractability.

Theorem 4 ($\star$). TDP *is in FPT when parameterized by the combination of the number $|S|$ of source-sink pairs and the feedback edge number of the underlying graph.*

The high-level idea of the algorithm is as follows. We can bound the number of paths in the underlying graph between any source-sink pair in a function of its feedback edge number. We can do the same for the number of how often two such paths intersect. Hence, for a given set of paths, the total number of such intersections is bounded by a function of the feedback edge number and the number of source-sink pairs. For each such intersection, we can consider all possibilities in which order it is traversed by the temporal paths. This gives us enough information to verify in polynomial time whether the possibility of how and in which order the source-sink pairs should be connected is realizable.

4.2 Algorithm for TDW

In this section, we present an FPT-algorithm for TDW parameterized by the number of source-sink pairs for the case where the underlying graph is a path. Recall that a temporal graph that has a path as underlying graph is called a *temporal line*. Klobas *et al.* [2023] showed that TDW is NP-hard on temporal lines and they gave an FPT-algorithm for TDP parameterized by the number of source-sink pairs for temporal forests. Theorem 3 implies that we presumably cannot adapt this FPT-algorithm for TDW. However, we can obtain tractability for the case of temporal lines. This answers an open question by Klobas *et al.* [2023].

Theorem 5. TDW *on temporal lines is in FPT with respect to the number $|S|$ of source-sink pairs.*

Before we prove Theorem 5, we first investigate properties of solutions $\mathcal{S}$ to an instance of TDW that minimize the sum of the lengths of its walks (our algorithm will produce such a solution). We show that we can upper-bound the number of times a temporal walk in such a solution $\mathcal{S}$ changes its direction by a function of $|S|$. Furthermore, we show that the direction changes always occur in "regions" (whose size is upper-bounded by a function of $|S|$) "around" the sources and sinks in S. Intuitively, this allows us to iterate over all possibilities in which direction, how often, and in which order the temporal walks move through the regions around the source and sink vertices in S. Given such a possibility, we have enough information to check whether there exist temporally disjoint walks that realize this behavior.

For the remainder of this section, let $\mathcal{S}$ be a solution to an instance of TDW that minimizes the sum of the lengths of its temporal walks. We first show that if a temporal walk in $\mathcal{S}$ changes its direction, there has to be another temporal walk in $\mathcal{S}$ that enforces this behavior as follows.

Lemma 6 ($\star$). *Let W be a temporal (s, z)-walk in $\mathcal{S}$ such that $(a, b, t), (b, a, t')$ with $t < t'$ are consecutive in W. Then, there exists a temporal (s', z')-walk W' in $\mathcal{S}$ with (c, a, t'') or (a, c, t'') in W' where $t < t'' < t'$.*

Having Lemma 6, we can inductively show if a temporal walk W_0 in $\mathcal{S}$ changes direction, then there is a sequence of temporal walks in $\mathcal{S}$ that change direction right before W_0 followed by a temporal walk in $\mathcal{S}$ that is either starting at its source or arriving at its sink.

Lemma 7 ($\star$). *Let W_0 be a temporal (s_0, z_0)-walk in $\mathcal{S}$ such that $(a_0, b_0, t_0), (b_0, a_0, t_0')$ with $t_0 < t_0'$ are consecutive in W_0. Then, there exist temporal (s_1, z_1)-walk $W_1, \dots,$ temporal (s_r, z_r)-walk W_r in $\mathcal{S}$ and $a_1, b_1, t_1, t_1', \dots, a_r, b_r, t_r, t_r'$ so that:*

[2]The *feedback edge number* of a graph G is the minimum number of edges that need to be removed from G to turn G into a forest.

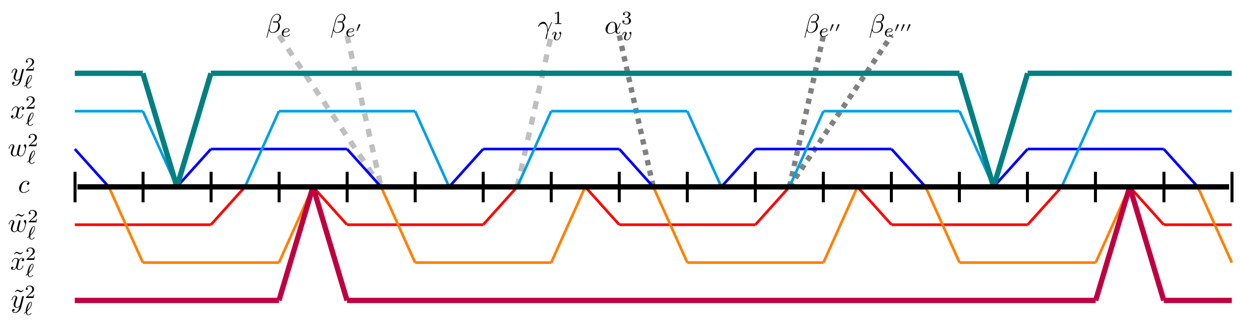

Figure 1: Illustration of the part of the temporal graph $\mathcal{G}$ as defined by Construction 1 that corresponds to color $i = 2$ for $k = 3$ colors. Vertices are represented by horizontal lines. The positions of the lines indicate the "vertex type" ($w_\ell^i, x_\ell^i, y_\ell^i, \tilde{w}_\ell^i, \tilde{x}_\ell^i, \tilde{y}_\ell^i$ for $\ell \in [n]$, and c), as listed on the left. The non-horizontal lines represent time edges, where the label corresponds to the position of their connection to the center vertex c (black horizontal line in the middle). Positions further to the left correspond to earlier time labels. Edges e, e' connect vertices of color 1 to vertex v of color 2. Edges e'', e''' connect vertices of color 3 to vertex v of color 2.

- *For every $1 \leq i < r$, $(a_i, b_i, t_i), (b_i, a_i, t'_i)$ are consecutive in W_i, $t_{i-1} < t_i \leq t'_i < t'_{i-1}$ and $a_{i-1} = b_i$.*
- *either (a_r, b_r, t_r) or (b_r, a_r, t_r) in W_r, $t_{r-1} < t_r < t'_{r-1}$, $a_{r-1} = b_r$ and either $b_r = z_r$ or $b_r = s_r$.*

From Lemma 7 we can draw two important corollaries that will help us to design the algorithm and prove its correctness. We give an illustration in Figure 2. The first corollary is that direction changes of temporal walks in $\mathcal{S}$ occur not too far away from source or sink vertices.

Corollary 8. *Let W be an (s, z)-walk in $\mathcal{S}$ such that $(a, b, t), (b, a, t')$ are consecutive in W with $t < t'$. Then, there exists an (s', z')-walk $W' \neq W$ in $\mathcal{S}$ with at least one among (s', a', t'') or (a', z', t'') in W' for some a', where $t < t'' < t'$ and the distance between a and a' in the underlying path of the temporal graph is at most $|S|$.*

The second corollary is that the temporal walks in $\mathcal{S}$ do not change their direction too often.

Corollary 9. *Let W be an (s, z)-walk in $\mathcal{S}$. Then at most $2|S|$ pairs of triples of the form $(a, b, t), (b, a, t')$ with $t < t'$ are consecutive in W.*

We now have all the pieces we need to prove Theorem 5.

Proof of Theorem 5. Let $(\mathcal{G} = (V, (E_t)_{t\in[T]}), S)$ be an instance of TDW such that the underlying graph of $\mathcal{G}$ is a path. Let G denote the underlying graph of $\mathcal{G}$. Recall that $\hat{S}$ denotes the set of all vertices in V that appear as sources or sinks in S. We know by Corollary 8 that we may assume w.l.o.g. that all temporal walks in a solution to $(\mathcal{G}, S)$ change direction only at vertices that are of distance (in G) at most $|S|$ from some vertex $v \in \hat{S}$. Let $D = \{v \in V \mid \exists v' \in \hat{S}$ such that $\text{dist}_G(v, v') \leq |S|\}$ denote the set of all vertices in V where some temporal walk in the solution potentially changes direction. Observe that $|D| \leq 4|S|^2$. By Corollary 9, we know that w.l.o.g. all temporal walks in a solution to $(\mathcal{G}, S)$ change direction at most $2|S|$ times. It follows that for each source-sink pair in $(s, z) \in S$, there are $|S|^{O(|S|)}$ possibilities we need to consider for where the temporal (s, z)-walk in the solution changes directions. Considering all source-sink pairs, we have $|S|^{O(|S|^2)}$ possible configurations the we need to consider for where temporal walks in the solution change directions.

Consider one specific configuration. We now analyse how many different relative orderings of the temporal walks we need to consider. To do this, we treat every temporal walk as at most $2|S|$ temporal path segments that form the walk, that is, at the endpoints of each path segment, the walk changes direction (or starts/ends). In total, this gives us $2|S|^2$ temporal path segments. Note that any two of these path segments P, P' have the property that they either do not visit common vertices, or if they do, then for all common vertices we have that either P occupies each of them before P' or vice versa. Otherwise, P and P' would be temporally intersecting. It follows that there exist a total ordering of all path segments such that whenever two path segments visit common vertices, the ordering defines which of the two path segments occupies each of the common vertices first. Overall, we have $|S|^{O(|S|^2)}$ possible orderings for the path segments.

However, some of these orderings might not yield pairwise temporally disjoint walks when we reconnect all path segments to form the respective temporal walks. Let P_1 and P_2 be two consecutive path segments of some temporal walk W such that P_1 is the path segment right before P_2, then the ordering must obey two requirements.

1. Path segment P_1 must occur before path segment P_2 in the ordering.
2. Let vertex v be the endpoint of P_1 and the starting point of P_2, implying that W changes direction at vertex v. Then for each path segment P' that contains vertex v we have that P' either must be before P_1 and P_2 in the ordering or P' must be after P_1 and P_2 in the ordering.

The first requirement must be met, since otherwise connecting P_1 and P_2 does not yield a temporal walk. To see why the second requirement must be met, let W' be the temporal walk of which P' is a path segment. If $W = W'$, then the first requirement is not met. If $W \neq W'$, then the two temporal walks would temporally intersect in vertex v.

We call an ordering of the path segments *valid* if both of the above requirements are met. Given an ordering, we can

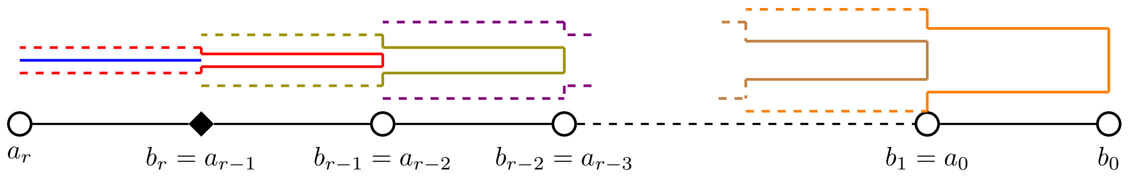

Figure 2: Illustration for Lemma 7 and Corollary 8. The colored paths represent temporal (s_r, z_r)- to (s_0, z_0)-walks. The vertical position of the colored edges represents the time labels. The diamond shaped vertex $b_r = a_{r-1}$ equals s_r or z_r and hence is a vertex of interest. Since $r \leq |S|$, the distance between b_r and b_0 is also at most $|S|$.

clearly check in polynomial time whether it is valid or not.

The algorithm now proceeds as follows:

1. Iterate over all possible configurations where the temporal walks in the solution change directions.
2. For each configuration, iterate over all valid orderings of the path segments implicitly given by the configuration.
3. For each configuration with a valid ordering, iterate over the path segments according to the ordering.

   For each path segment, compute a prefix-foremost temporal path P from the starting point to the endpoint of the path segment.

   If no such temporal path exists, discard the current combination of configuration with valid ordering. Otherwise, for each transition (v, w, t) in P, remove all time edges incident with v or w that have a time label $t' \leq t$. Continue with the next path segment. If there is no further path segment, output YES.
4. If all combinations of configuration with valid ordering were discarded, output NO.

Since we have $|S|^{O(|S|^2)}$ possible configurations and $|S|^{O(|S|^2)}$ possible valid orderings, the running time of the algorithm is in $|S|^{O(|S|^4)} \cdot |\mathcal{G}|^{O(1)}$. Note that with polynomial overhead, the algorithm can also output the solution.

By the arguments made before, it is easy to check that if the algorithm outputs YES, then we face a yes-instance.

For the other direction, assume that we face a yes-instance. Then there is a solution $\mathcal{S}$ that minimizes the sum of the lengths of its temporal walks. Corollaries 8 and 9 imply that each temporal walk in the solution changes direction at most $2|S|$ times at vertices that are of distance at most $|S|$ to a vertex that appears as a source or sink in S. Hence, we can segment every temporal walk in the solution into at most $2|S|$ temporal paths that have endpoints in the set D (defined at the beginning of the proof). The path segments form a partially ordered set, if we define a path segment P to be smaller than P' if the two path segments have common vertices and each of the common vertices is occupied by P earlier than by P'. Note that for all pairs of path segment P, P' that have common vertices, we have that P is either smaller than P' or vice versa, otherwise P and P' would be temporally intersecting. Hence, we have that any linearization of the partial ordering is a valid ordering of the path segments.

If follows that there exists a combination of configurations with valid ordering that agrees with the solution. Lastly, note that we can assume w.l.o.g. that the temporal path segments in the solution are prefix-foremost (among the ones that do not temporally intersect), since if they are not, we can simply replace a temporal path segment with a prefix-foremost one. We can conclude that the algorithm outputs YES. □

5 Future Work

We leave several directions for future research. Our hardness results rule out many structural graph parameters as further options for obtaining tractability. However, there are some candidates that are unrelated to the vertex cover number and the feedback edge number, and are also large on star graphs, leading to the following question.

- Are TDP and TDW in FPT or W[1]-hard with respect to the combination of the number of source-sink pairs and the *cutwidth* or *bandwidth* of the underlying graph?

In MAPF, one is often interested in finding solutions that minimize the sum or maximum of steps or actions that each agent needs to take to arrive at their destination [Stern, 2019]. In our setting, the number of transitions of a temporal path or walk corresponds to the number of steps and the difference between the time label of the last and first transition (also called *duration*) corresponds to the number of actions (where waiting for one time step is considered an action). We can observe that the number of transitions of temporal paths or walks is constant in the reductions of Theorem 1 and Theorem 2, indicating that finding solutions with few transitions is still hard. We believe that the duration might be a more promising parameter, since it is large in the reductions of Theorem 2 and Theorem 3, leading to the following question.

- Are TDP and TDW in FPT or W[1]-hard with respect to the combination of the number of source-sink pairs and the maximum duration of any temporal (path/walk) in the solution?

Finally, we leave open whether our results also hold for the non-strict case, where temporal (paths/walks) use transitions with non-decreasing (instead of increasing) time labels. We conjecture that all our results can be adapted for this case.

Acknowledgments

Pascal Kunz was supported by the DFG, Research Training Group 2434 “Facets of Complexity”. Hendrik Molter was supported by the ISF, grant No. 1456/18, and European Research Council, grant number 949707.

References

[Almagor and Lahijanian, 2020] Shaull Almagor and Morteza Lahijanian. Explainable multi agent path finding. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '20)*, pages 34–42, 2020.

[Andreychuk *et al.*, 2022] Anton Andreychuk, Konstantin Yakovlev, Pavel Surynek, Dor Atzmon, and Roni Stern. Multi-agent pathfinding with continuous time. *Artificial Intelligence*, 305:103662, 2022.

[Atzmon *et al.*, 2020] Dor Atzmon, Roni Stern, Ariel Felner, Glenn Wagner, Roman Barták, and Neng-Fa Zhou. Robust multi-agent path finding and executing. *Journal of Artificial Intelligence Research*, 67:549–579, 2020.

[Casteigts *et al.*, 2021] Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, and Philipp Zschoche. Finding temporal paths under waiting time constraints. *Algorithmica*, 83(9):2754–2802, 2021.

[Cygan *et al.*, 2015] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[Downey and Fellows, 2013] Rodney G Downey and Michael R Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.

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

[Flum and Grohe, 2006] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Springer, 2006.

[Fortune *et al.*, 1980] Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980.

[Füchsle *et al.*, 2022] Eugen Füchsle, Hendrik Molter, Rolf Niedermeier, and Malte Renken. Delay-robust routes in temporal graphs. In *Proceedings of the 39th International Symposium on Theoretical Aspects of Computer Science (STACS)*, 2022.

[Goldreich, 2011] Oded Goldreich. Finding the shortest move-sequence in the graph-generalized 15-puzzle is NP-hard. In *Studies in complexity and cryptography. Miscellanea on the interplay between randomness and computation*, pages 1–5. Springer, 2011.

[Holme and Saramäki, 2019] Petter Holme and Jari Saramäki. *Temporal Network Theory*. Springer, 2019.

[Jansen *et al.*, 2013] Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences*, 79(1):39–49, 2013.

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

[Kawarabayashi *et al.*, 2012] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012.

[Kempe *et al.*, 2002] David Kempe, Jon Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002.

[Klobas *et al.*, 2023] Nina Klobas, George B. Mertzios, Hendrik Molter, Rolf Niedermeier, and Philipp Zschoche. Interference-free walks in time: temporally disjoint paths. *Autonomous Agents and Multi-Agent Systems*, 37(1):1, 2023.

[Korte *et al.*, 1990] Bernhard Korte, László Lovász, Hans Jürgen Prömel, and Alexander Schrijver. *Paths, flows, and VLSI-layout*. Springer, 1990.

[Kunz *et al.*, 2023] Pascal Kunz, Hendrik Molter, and Meirav Zehavi. In which graph structures can we efficiently find temporally disjoint paths and walks? *CoRR*, abs/2301.10503, 2023.

[Latapy *et al.*, 2018] Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, 8(1):61:1–61:29, 2018.

[Li *et al.*, 2019] Jiaoyang Li, Pavel Surynek, Ariel Felner, Hang Ma, TK Satish Kumar, and Sven Koenig. Multi-agent path finding for large agents. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI '19)*, pages 7627–7634, 2019.

[Ma *et al.*, 2017] Hang Ma, TK Satish Kumar, and Sven Koenig. Multi-agent path finding with delay probabilities. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI '17)*, pages 3605–3612, 2017.

[Michail, 2016] Othon Michail. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, 12(4):239–280, 2016.

[Neil and Seymour, 1985] Robertson Neil and Paul D. Seymour. Disjoint paths—a survey. *SIAM Journal on Algebraic and Discrete Methods*, 6(2):300–305, 1985.

[Neil and Seymour, 1995] Robertson Neil and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

[Salzman and Stern, 2020] Oren Salzman and Roni Stern. Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AMAAS '20)*, pages 1711–1715, 2020.

[Sharon *et al.*, 2015] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

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

[Standley, 2010] Trevor Scott Standley. Finding optimal solutions to cooperative pathfinding problems. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI '10)*, pages 173–178, 2010.

[Stern *et al.*, 2019] Roni Stern, Nathan R Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Eli Boyarski, and Roman Barták. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the 12th International Symposium on Combinatorial Search (SOCS)*, pages 151–159, 2019.

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

[Švancara *et al.*, 2019] Jiří Švancara, Marek Vlk, Roni Stern, Dor Atzmon, and Roman Barták. Online multi-agent pathfinding. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI '19)*, pages 7732–7739, 2019.

[Wu *et al.*, 2016] Huanhuan Wu, James Cheng, Yiping Ke, Silu Huang, Yuzhen Huang, and Hejun Wu. Efficient algorithms for temporal path computation. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2927–2942, 2016.

[Yu and LaValle, 2013] Jingjin Yu and Steven M LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI '13)*, pages 1443–1449, 2013.