Algorithms and complexity for path covers of temporal DAGs

Dibyayan Chakraborty, Antoine Dailly, Florent Foucaud, Ralf Klasing

Algorithms and complexity for path covers of temporal DAGs: when is Dilworth dynamic?*

Dibyayan Chakraborty[1], Antoine Dailly[2], Florent Foucaud[2], and Ralf Klasing[3]

[1]School of Computing, University of Leeds, United Kingdom
[2]Université Clermont-Auvergne, CNRS, Mines de Saint-Étienne, Clermont-Auvergne-INP, LIMOS, 63000 Clermont-Ferrand, France
[3]Université de Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR 5800, Talence, France

Abstract

A *path cover* of a digraph is a collection of paths collectively containing the vertex set of the digraph. A path cover with minimum cardinality for a *directed acyclic graph* can be found in polynomial time [Fulkerson, AMS '56; Cáceres et al., SODA'22]. Moreover, Dilworth's celebrated theorem on chain coverings of partially ordered sets equivalently states that the minimum size of a path cover of a DAG is equal to the maximum size of a set of mutually unreachable vertices. In this paper, we examine how far Dilworth's theorem can be extended to a "*dynamic*" analogue of directed acyclic graphs.

A temporal digraph has an arc set that changes over discrete time-steps. Furthermore, if the underlying digraph (*i.e.*, the union of all the arc sets that appears at some point) is acyclic, then we have a temporal directed acyclic graph (or simply a temporal DAG). A temporal path is a directed path in the underlying digraph, such that the time-steps of arcs are strictly increasing along the path. Two temporal paths are temporally disjoint if they do not occupy any vertex at the same time. A *temporal path cover* is a collection $\mathcal{C}$ of temporal paths that covers all vertices. Furthermore, $\mathcal{C}$ is a *temporally disjoint path cover* if all temporal paths are pairwise temporally disjoint. In this paper, we study the computational complexities of the problems of finding a temporal (disjoint) path cover with minimum cardinality (denoted as Temporal Path Cover and Temporally Disjoint Path Cover).

We show that both Temporal Path Cover and Temporally Disjoint Path Cover are NP-hard even when the underlying DAG is planar, bipartite, subcubic, and there are only two arc-disjoint time-steps. Moreover, Temporally Disjoint Path Cover remains NP-hard even on temporal oriented trees. We also observe that natural temporal analogues of Dilworth's theorem on these classes of temporal DAGs do not hold.

In contrast, we show that Temporal Path Cover is polynomial-time solvable on temporal oriented trees by a reduction to Clique Cover for (static undirected) *weakly chordal* graphs (a subclass of perfect graphs for which Clique Cover admits an efficient algorithm). This highlights an interesting algorithmic difference between the two problems. Although it is NP-hard on temporal oriented trees, Temporally Disjoint Path Cover becomes polynomial-time solvable on temporal oriented lines and temporal *rooted directed* trees. For all these positive algorithmic results, we also show that temporal analogues of Dilworth's theorem hold for the corresponding temporal graph classes.

We also show that Temporal Path Cover and Temporally Disjoint Path Cover become efficiently solvable when the number of time-steps is bounded *and* the underlying graph is close to a tree. More precisely, we show that Temporal Path Cover admits an XP time algorithm with respect to parameter $t_{\max} + tw$, where $t_{\max}$ is the maximum time-step, and tw is the treewidth of the underlying static undirected graph. We also show that Temporally Disjoint Path Cover admits an FPT algorithm with respect to the same parameter.

*This work was supported by the International Research Center "Innovation Transportation and Production Systems" of the I-SITE CAP 20-25 and by the ANR project GRALMECO (ANR-21-CE48-0004). Ralf Klasing's research was partially supported by the ANR project TEMPOGRAL (ANR-22-CE48-0001).

1 Introduction

A classic theorem of Dilworth from 1950 [13] states that in any partially ordered set (poset), the minimum number of chains required to cover all the elements is equal to the maximum size of an antichain. Dilworth's theorem is fundamental from the mathematical point of view; furthermore, an algorithmic proof (that enables to construct a chain cover and an antichain in polynomial time) was published by Fulkerson in 1956 [16]. This theorem and its algorithmic form have many applications, not only in combinatorics, but also in various fields such as bioinformatics [6], scheduling [33], databases [22], program testing [38], etc.

A collection $\mathcal{P}$ of (resp. pairwise vertex-disjoint) directed paths of a digraph D is a *path cover* (resp. *path partition*) of D if all vertices of D are contained in some path of $\mathcal{P}$. Dilworth's theorem can be restated in an equivalent form, equating the minimum cardinality of path covers on directed acyclic graphs (DAGs) and the maximum size of a set of pairwise "unreachable" vertices, or *antichain* vertices [4, 5, 15].

Theorem 1 (Dilworth [13])**.** *For any DAG D, the minimum number of paths that cover its vertex set, is equal to the maximum size of an antichain of D.*

Fulkerson [16] showed that finding a minimum-size path cover of a DAG can be done in polynomial time. Moreover, it is known that finding a minimum-size path partition can also be done in polynomial time for arbitrary DAGs [10, Probl. 26-2]. Improving the best known algorithms for path cover and partitions of DAGs still form an active field of research, see for example [4, 5, 9, 32] for some recent results.

The notions of directed paths and path covers naturally extends to *temporal (di)graphs*. Informally, the arc set of a temporal digraph changes over discrete time-steps and *labels* of an arc are the time-steps where the arc appears. Temporal (di)graphs have been extensively studied in the two last decades, with contributions from and applications to various fields, see [7, 21, 23, 35, 36, 39]. A *temporal path* of a digraph is a path that traverses edges appearing at strictly increasing time-steps. The asymmetric nature of temporal paths has motivated many recent algorithmic works on related reachability or path problems on temporal graphs, such as [1, 2, 3, 8, 24, 34].

Two temporal paths are *temporally disjoint* if they do not occupy a vertex at the same time-step. This definition was introduced by Klobas et al. [25] and has since then garnered attention in the graph algorithmic community [29]. Even though the above notion was introduced in the context of temporal undirected graphs, it naturally extends to temporal digraphs and motivates the corresponding covering problems. The objective of Temporal Path Cover (resp. Temporally Disjoint Path Cover) is to cover an input temporal digraph by a minimum number of temporal paths (resp. temporally disjoint paths).

Main objectives. In this paper, we initiate the algorithmic study of Temporal Path Cover and Temporally Disjoint Path Cover and focus on *temporal directed acyclic graphs* (or simply, temporal DAGs). A temporal digraph is a *temporal DAG* if the union of all arcs across all time-steps induces a (static) DAG. We say that a temporal digraph satisfies the *Dilworth property* (resp. *temporally disjoint Dilworth property*, or *TD-Dilworth property* for short) if the largest size of a *temporal antichain* (understood as a set of pairwise unreachable vertices) is equal to the smallest size of a temporal path cover (resp. temporally disjoint path cover). The main goals of this paper are the following:

(a) Determine classes of temporal DAGs satisfying the (TD-)Dilworth property.

(b) Study the computational complexities of Temporal Path Cover and Temporally Disjoint Path Cover on temporal digraphs.

Practical motivations. A first motivation is multi-agent-based decision-making (a well-studied problem from artificial intelligence [41, 44]) in a temporal setting, such as for coral reef protection [45] or crime prevention in transportation networks [46]. In this setting, the temporal DAG can model a decision-making process, where the vertices represent the states of an environment. Agents navigate the DAG, an arc representing an agent's move from one state to another. As the situation is varying over time, a move may only be available at specific time-steps. A path in this DAG thus represents the overall activity of an agent. In this setting, Temporal Path Cover represents the situation where a set of k agents need to cover all the possible states.

In Temporally Disjoint Path Cover, the agents must also avoid each other, and cannot cover the same state at the same time, a scenario described as *vertex-conflicts* in the literature [41].

Another natural application is *multi-robot path planning* [12, 42]. Imagine the setting where k robots are assigned the task of exploring a hazardous facility. Since the facility changes over time, it is modeled as a temporal digraph. If the facility digraph does not contain directed cycles, it is modeled by a temporal DAG (for example, if the facility is inherently directed from a start area towards a target area). The exploration path of a robot can be modeled by a temporal path. Now, Temporal Path Cover corresponds to the situation where the robots need to explore the whole facility, while for Temporally Disjoint Path Cover, the robots also cannot be simultaneously at the same location.

Our results. We begin by formally defining the problems studied in this paper.

Temporal Path Cover (TPC)
Input: A temporal digraph D, an integer k.
Problem: Does there exist a set $\mathcal{C}$ of k temporal paths in D such that every vertex of D is covered by some path of $\mathcal{C}$?

Temporally Disjoint Path Cover (TD-PC)
Input: A temporal digraph D, an integer k.
Problem: Does there exist a set $\mathcal{C}$ of k temporally disjoint temporal paths in D such that every vertex of D is covered by some path of $\mathcal{C}$?

We observe that in general, temporal DAGs do not have the Dilworth property (see Figure 1a). Then, we prove the following negative result.

Theorem 2. *Temporal Path Cover and Temporally Disjoint Path Cover are NP-hard on temporal DAGs, even if the input is planar, bipartite, subcubic, of girth 10, uses only one time label per arc, and every label is either* 1 *or* 2.

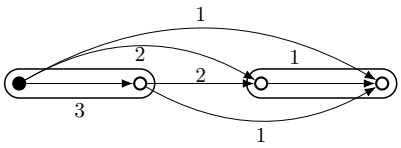
A temporal directed acyclic graph D is a *temporal oriented tree* if the underlying directed graph of D is a tree. On the positive side, we prove the following.

Theorem 3. *There is an $\mathcal{O}(\ell n^2 + n^3)$-time algorithm for Temporal Path Cover on temporal oriented trees with n vertices and at most ℓ many labels per arc. Furthermore, temporal oriented trees satisfy the Dilworth property.*

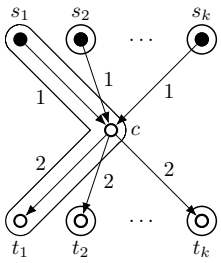
We briefly describe the technique we use for proving Theorem 3. Two vertices of a temporal digraph are *temporally connected* if they are covered by the same temporal path. The *connectivity graph* of a temporal digraph D is an undirected (static) graph whose vertex set is the same as that of D, and whose edge set consists of all pairs of temporally connected vertices. To prove the above theorem, we show that the connectivity graph of a temporal oriented tree is a *weakly chordal graph* [19] (a subclass of *perfect graphs*). We show Temporal Path Cover can be reduced to Clique Cover on weakly chordal graphs. The above observation, combined with the Weak Perfect Graph Theorem (proved by Lovász [31]), proves that temporal oriented trees satisfy the Dilworth property. Moreover, the existing $\mathcal{O}(nm)$-time algorithm [20] to compute a minimum clique cover of a weakly chordal graph (having n vertices and m edges) completes the proof of Theorem 3. Our proof gives interesting structural information on the interaction between temporal paths in temporal oriented trees. Interestingly, another important class of perfect graphs plays an important role in connection with Dilworth's theorem and its translation to the setting of static DAGs: the class of comparability graphs, see [18, Chapter 5.7]. In our case, there does not appear to be any connection to comparability graphs.

On the other hand, temporal oriented trees do not satisfy the TD-Dilworth property (see Figure 1b for an example). Then, we prove the following negative result.

Theorem 4. *Temporally Disjoint Path Cover is NP-hard on temporal oriented trees.*



(a) A temporal DAG not having the Dilworth property.



(b) A temporal oriented tree not having the TD-Dilworth property.

Figure 1: A minimum-size (temporally disjoint) temporal path cover is shown, vertices in a maximum-size temporal antichain are in black.

To find classes that satisfy the TD-Dilworth property, we study *temporal oriented lines* (that is, where the underlying digraph is an oriented path) and *temporal rooted directed trees*. A tree is a *rooted directed tree* if it is an oriented tree with a single source vertex called the *root*. We prove the following result.

Theorem 5. *Temporal Path Cover and Temporally Disjoint Path Cover can be solved in time:*

(a) $\mathcal{O}(\ell n)$ *on temporal oriented lines;*

(b) $\mathcal{O}(\ell n^2)$ *on temporal rooted directed trees;*

where ℓ *is the maximum number of labels per arc and* n *is the number of vertices. Furthermore, both classes satisfy the TD-Dilworth property.*

Note that some related problems remain NP-hard for temporal lines, such as Temporally Disjoint Walks [26]. Theorem 5(a) shows that this is not the case here. To prove Theorem 5(b), we begin by constructing a temporal path cover before transforming it into a temporally disjoint one of the same size. This is in contrast with general temporal oriented trees, for which, by Theorem 4, such an approach is not possible.

As Temporally Disjoint Path Cover is NP-hard even on temporal oriented trees and on temporal DAGs with two time-steps, a natural question is what happens when the number of time-steps is small *and* the underlying digraph is a tree. Motivated by this question, we study the case where both the number of time-steps and the treewidth of the underlying digraph are bounded (where we define the *treewidth* of a temporal digraph as the treewidth of the underlying static undirected graph). We show that both problems become tractable in this setting. More precisely, we give a fixed-parameter tractable (FPT) algorithm for Temporally Disjoint Path Cover with treewidth and number of time-steps as parameters. The same technique gives an XP algorithm for Temporal Path Cover.

Theorem 6. *There is an algorithm for Temporally Disjoint Path Cover on general temporal digraphs that is FPT with respect to the treewidth of the underlying undirected graph and the maximum number of labels per arc. For Temporal Path Cover on general temporal digraphs, there is an XP algorithm for the same parameter.*

See Table 1 for a summary of our algorithmic results.

Further related work. Algorithms for solving several types of path and distance problems in temporal graphs have been developed, see for example [3, 24, 43]. Recently, the problem Temporally Disjoint Paths was introduced in [25], as a generalization of the notorious Disjoint Paths problem (also known as Linkage). In Temporally Disjoint Paths, one is given a temporal graph with k pairs of vertices called *terminals*, and the goal is to find a set of k pairwise temporally disjoint paths, each of them connecting one pair of terminals. Temporally Disjoint Paths is NP-hard, even for temporal lines and two paths [25] or temporal stars [29], but becomes FPT for trees when parameterized by the number of paths [25]. Algorithms that are FPT for certain structural parameters are given in [29].

temporal graph class	TPC	TD-PC
temporal DAGs (planar bipartite subcubic, girth 10, two arc-disjoint time-steps)	NP-c.	NP-c.
temporal oriented trees	poly	NP-c.
temporal rooted directed trees	poly	poly
temporal oriented lines	poly	poly
general temporal digraphs with bounded treewidth and number of time-steps	poly (XP)	poly (FPT)

Table 1: Summary of our algorithmic results. For all polynomial-time solvable classes of temporal DAGs, we also show that the Dilworth property (or TD-Dilworth property for TD-PC) holds.

Structure of the paper. We start with the hardness result for temporal DAGs (Theorem 2) in Section 3. We then prove our results for temporal oriented trees (Theorem 3 and Theorem 4) in Sections 4 and 5. We prove Theorem 5, the polynomial-time algorithms for special temporal oriented trees (temporal rooted directed trees and temporal oriented lines), in Section 6. We then prove our results for temporal digraphs of bounded treewidth and number of time-steps (Theorem 6) in Section 7. We conclude in Section 8.

2 Preliminaries

A *temporal digraph* $\mathcal{D} = (V, A_1, \ldots, A_{t_{\max}})$ is given by a sequence of arc-sets representing $t_{\max}$ discrete *time-steps* $\{1, \ldots, t_{\max}\}$, where an arc in A_i is *active* at time-step i [25]. Let us denote by $D = (V, A)$, where $A = \cup_{i=1}^{t_{\max}} A_i$, the *underlying digraph* of temporal digraph $\mathcal{D} = (V, A_1, \ldots, A_{t_{\max}})$ (sometimes called *footprint (di)graph* [7]). Equivalently, one can view the time-steps as an arc-labelling function $\lambda : A(D) \to 2^{[t_{\max}]}$, where $\lambda(\overrightarrow{xy}) \subseteq [t_{\max}]$ is the set of time-steps where $\overrightarrow{xy}$ is active [24]. In that case, we may denote the temporal digraph as $\mathcal{D} = (D, \lambda)$. We say that a temporal digraph has a given property $\mathcal{P}$ (planarity, given girth...) if the undirected graph obtained by forgetting the orientation of the arcs of its underlying digraph has property $\mathcal{P}$. For a given temporal digraph, we denote by ℓ the maximum number of labels per arc and by n the number of vertices in the underlying digraph.

For a (temporal) (di)graph $\mathcal{D}$ and subset S of its vertices (resp. edges), $\mathcal{D} \setminus S$ denotes the (temporal) (di)graph obtained by removing the vertices (resp. edges) in S from $\mathcal{D}$.

In a temporal digraph, a *temporal (directed) path* is a sequence $(v_1, v_2, t_1), (v_2, v_3, t_2), \ldots, (v_{k-1}, v_k, t_{k-1})$ such that for any i, j with $1 \leq i < j \leq k$, $v_i \neq v_j$ and for any i with $1 \leq i \leq k-1$, $t_i < t_{i+1}$ and there is an arc $\overrightarrow{v_i v_{i+1}}$ at time-step t_i. These paths are sometimes called *strict* in the literature.[1] For a temporal path $P = (v_1, v_2, t_1), \ldots, (v_{k-1}, v_k, t_{k-1})$, we denote by $V(P)$ the set $\cup_{i=1}^{k} \{v_i\}$ and by $A(P)$ the set $\cup_{i=1}^{k-1} \{\overrightarrow{v_i v_{i+1}}\}$.

The *length* of a temporal path is the number of arcs it uses. We say that a temporal path $P = (v_1, v_2, t_1), \ldots, (v_{k-1}, v_k, t_{k-1})$ *occupies* vertex v_i during the time interval $\{t_{i-1}, \ldots, t_i\}$. Two temporal paths P_1, P_2 are *temporally disjoint* if for all arcs $e_1 \in A(P_1), e_2 \in A(P_2)$ incident with a common end-vertex, the time-step of e_1 in P_1 is distinct from the time-step of e_2 in P_2. In other words, two paths are temporally disjoint if they do not occupy the same vertex at the same time. A *temporal path cover* (resp. *temporally disjoint path cover*) of a temporal digraph D is a collection of temporal paths (resp. temporally disjoint paths) that cover all vertices of D. Two vertices are *temporally connected* in D if there exists a temporal path between them. A *temporal antichain* is a set of vertices that are pairwise not temporally connected.

Definition 7. *A class $\mathcal{C}$ has the* Dilworth property *(resp.* TD-Dilworth property*) if the cardinality of the minimum temporal path cover (resp. temporally disjoint path cover) is equal to the maximum cardinality of a temporal anti-chain.*

[1] For non-strict paths, the condition $t_i < t_{i+1}$ is replaced with $t_i \leq t_{i+1}$, but as argued in [29], the strict definition is more natural for applications where an agent cannot traverse an arbitrary number of arcs at once, this is why we chose this convention.

A *hole* of a static undirected graph is an induced cycle of length at least 5, and an *anti-hole* is the complement of a hole. A graph G is *weakly chordal* if it has no hole or anti-hole. A *(minimum) clique cover* of a graph G is a (minimum cardinality) set of complete subgraphs of G that covers all vertices. A *(maximum) independent set* of a graph G is a (maximum cardinality) set of pairwise non-adjacent vertices. We shall use the following results for weakly chordal graphs.

Theorem 8 ([20, 31, 40]). *Let H be a weakly chordal graph with n vertices and m edges. Then, a minimum clique cover of H can be found in $\mathcal{O}(nm)$-time. Furthermore, the maximum size of an independent set of H equals the minimum size of a clique cover of H.*

3 Temporal DAGs

We provide a reduction (inspired by [37]) from a restricted variant of 3-Dimensional Matching to prove the following.

Theorem 2. *Temporal Path Cover and Temporally Disjoint Path Cover are NP-hard on temporal DAGs, even if the input is planar, bipartite, subcubic, of girth 10, uses only one time label per arc, and every label is either 1 or 2.*

Proof. We will reduce the Temporal (Disjoint) Path Cover problem on temporal DAGs from the 3-Dimensional Matching problem. The reduction is inspired from [37].

> 3-Dimensional Matching (3DM)
> Instance: A set $S \subseteq X \times Y \times Z$, where X, Y, and Z are disjoint sets having the same number q of elements.
> Question: Does S contain a *perfect matching*, *i.e.*, a subset $M \subseteq S$ such that $|M| = q$ and no two elements of M agree in any coordinate?

It is well-known that 3-Dimensional Matching is NP-hard [17].

Given an instance $I = (S, X \times Y \times Z)$ of 3DM, where $S = \{s_1 \ldots, s_p\}$, $X = \{x_1, \ldots, x_q\}$, $Y = \{y_1, \ldots, y_q\}$ and $Z = \{z_1, \ldots, z_q\}$, we build an instance $\mathcal{D} = (V, A_1, A_2)$ of Temporal (Disjoint) Path Cover, where $\mathcal{D}$ is a temporal DAG, as follows.

To each triple $s_i = (x_{i,1}, y_{i,2}, z_{i,3}) \in S$, we associate a gadget $H(s_i)$ that consists of a collection $\{P^{i,1}, P^{i,2}, P^{i,3}\}$ of 3 directed vertex-disjoint paths of 3 vertices with $P^{i,r} = \{\overrightarrow{a_1^{i,r}a_2^{i,r}}, \overrightarrow{a_2^{i,r}a_3^{i,r}}\}$ for $r = 1, 2, 3$; and the time labels are 1 for the arcs $\overrightarrow{a_1^{i,r}a_2^{i,r}}$ and 2 for the arcs $\overrightarrow{a_2^{i,r}a_3^{i,r}}$. We add to $H(s_i)$ the arcs $\overrightarrow{a_3^{i,1}a_3^{i,2}}$ and $\overrightarrow{a_3^{i,2}a_3^{i,3}}$, in order to form a 4th directed path of 3 vertices; and the time labels are 1 for the arcs $\overrightarrow{a_3^{i,1}a_3^{i,2}}$ and 2 for the arcs $\overrightarrow{a_3^{i,2}a_3^{i,3}}$. Finally, we add to $H(s_i)$ the arcs $\overrightarrow{a_2^{i,1}x_{i,1}}$, $\overrightarrow{a_2^{i,2}y_{i,2}}$ and $\overrightarrow{a_2^{i,3}z_{i,3}}$, with the time label 2 (see Figure 2 for an illustration).

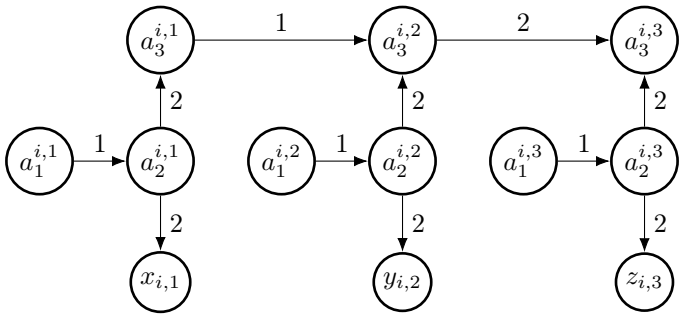
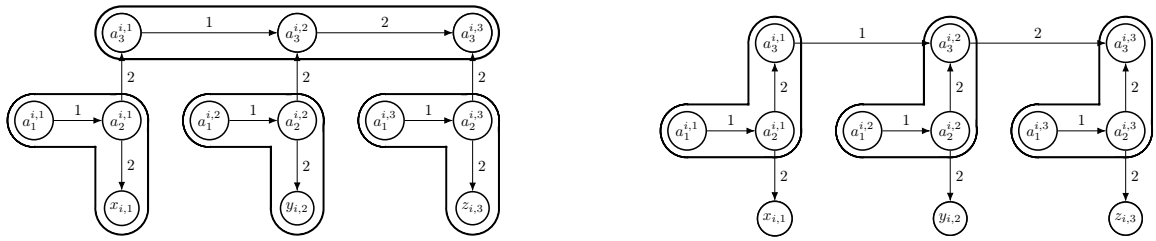


Figure 2: The gadget $H(s_i)$.



(a) $s_i \in M$ (b) $s_i \notin M$

Figure 3: Vertex partition of the gadget $H(s_i)$ into length-2 paths.

The above construction yields a temporal digraph $\mathcal{D}$ on $9p + 3q$ vertices. Note that the construction uses only 1 label per arc, and every label is either 1 or 2.

We claim that there exists a perfect matching $M \subseteq S$ in I if and only if there exists a temporal (disjoint) path cover (partition) of $\mathcal{D}$ of size $3p + q$.

($\Rightarrow$) Let $M \subseteq S$ be a perfect matching in S, and consider the following collection of directed vertex-disjoint temporal length-2 paths in the gadget $H(s_i)$: $\{\overrightarrow{a_3^{i,1}a_3^{i,2}}, \overrightarrow{a_3^{i,2}a_3^{i,3}}\}$, $\{\overrightarrow{a_1^{i,1}a_2^{i,1}}, \overrightarrow{a_2^{i,1}x_{i,1}}\}$, $\{\overrightarrow{a_1^{i,2}a_2^{i,2}}, \overrightarrow{a_2^{i,2}x_{i,2}}\}$, $\{\overrightarrow{a_1^{i,3}a_2^{i,3}}, \overrightarrow{a_2^{i,3}x_{i,3}}\}$ if $s_i \in M$, and $P^{i,1}, P^{i,2}, P^{i,3}$ if $s_i \notin M$. As M is a perfect matching in S, the collection of the temporal paths defined above constitute a vertex-disjoint (and thus temporally disjoint) temporal path cover of $\mathcal{D}$ of size $3p + q$. Figures 3a and 3b illustrate the construction of the temporal path partition on $V(H(s_i))$ with respect to a given matching M for 3DM.

($\Leftarrow$) Assume that there exists a (temporally disjoint) path cover $\mathcal{C}$ of $\mathcal{D}$ of size $3p+q$. As $|V(\mathcal{D})| = 9q+3p$, $|\mathcal{C}| = 3p + q$ and every (temporal) path in $\mathcal{D}$ has length at most 2, all paths in $\mathcal{C}$ must have exactly length 2, and $\mathcal{C}$ is indeed a *partition* of $V(\mathcal{D})$ into length-2 paths. All length-2 paths in $\mathcal{D}$ are depicted in Figures 3a and 3b. Hence, any path partition $\mathcal{C}$ of $\mathcal{D}$ must have, for each triple gadget, the path structure as depicted in either one of Figures 3a and 3b, and there must be q gadgets $H_1, \ldots, H_q$ that are each covered by four vertex-disjoint temporal length-2 paths from $\mathcal{C}$, and $p - q$ gadgets $H_{q+1}, \ldots, H_p$ where the vertices $a_1^{i,r}a_2^{i,r}a_3^{i,r}$ (for $r = 1, 2, 3$) are covered by three vertex-disjoint temporal length-2 paths from $\mathcal{C}$ and the vertices $x_{i,1}, x_{i,2}, x_{i,3}$ are not covered. Then, the triples $(x_{i,1}, y_{i,2}, z_{i,3})$ corresponding to $H_1, \ldots, H_q$ constitute a perfect matching in S.

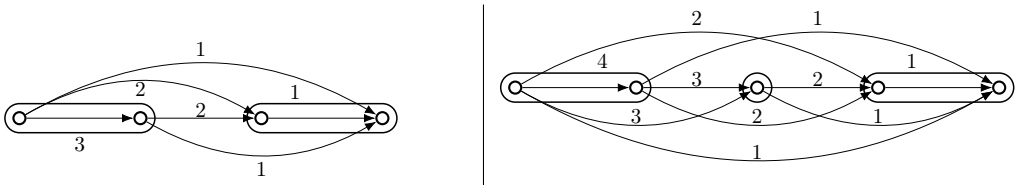
This completes the NP-hardness proof of Temporal (Disjoint) Path Cover in temporal DAGs. In order to show that Temporal (Disjoint) Path Cover remains NP-hard in planar, bipartite, subcubic temporal DAGs of girth 10, we apply the above proof, except that we start from a restriction of the three-dimensional matching problem, in which every element appears in either two or three triples, and the associated bipartite graph (formed by the elements and triples as its vertices, with edges connecting each element to the triples it belongs to) is planar and subcubic, denoted by Planar 3DM-3. It is well-known that this restriction of 3DM is still NP-hard [14]. Following a planar embedding of the bipartite graph associated to the instance of Planar 3DM-3, one can obtain a planar enmbedding of the constructed graph. Note that the underlying DAG in the above reduction is bipartite, as it can be 2-colored as follows: vertices in X and Z are colored 1, vertices in Y are colored 2, and then the coloring can be extended to the triple gadgets. Note as well that the shortest cycle in the underlying undirected graph has length 10. □

We also show the following.

Proposition 9. *There are temporal DAGs (whose underlying digraph is a transitive tournament) that satisfy neither the Dilworth nor the TD-Dilworth property. Moreover, the ratio between the minimum-size temporal path cover and temporally disjoint path cover and the maximum-size temporal antichain can be arbitrarily large.*

Proof. Consider the temporal digraph $\mathcal{T}_n = (T_n, \lambda)$ where T_n is the transitive tournament on vertices $u_1, \ldots, u_n$ and $\lambda(\overrightarrow{u_iu_j}) = n - (j + 1)$ for all $i < j$. $\mathcal{T}_n$ is a temporal DAG, and since its underlying digraph

is a transitive tournament, all the pairs of vertices are temporally connected, implying that the temporal antichain is of size 1. However, no temporal path can contain more than two vertices, and thus $\lceil \frac{n}{2} \rceil$ paths are needed to cover it. Hence, the gap between the maximum size of a temporal antichain and the minimum size of a temporal path cover can be as large as we want for a temporal DAG. Furthermore, the minimum-size TPC is also vertex-disjoint (and thus temporally disjoint) if n is even since the paths will never intersect; and if n is odd, then at most two paths in a minimum-size TPC can intersect in at most one vertex, thus we can reduce one of the two intersecting paths to cover only one vertex, giving us a TPC of the same size with vertex-disjoint, and thus temporally disjoint, paths. This is depicted in Figure 4. □



(a) $n = 4$ (b) $n = 5$

Figure 4: $\mathcal{T}_n$, a temporal DAG with a maximum-size temporal antichain of size 1 and a minimum-size temporal path cover of size $\lceil \frac{n}{2} \rceil$, for $n = 4, 5$.

4 Temporal Path Cover on temporal oriented trees

In this section we prove the following theorem.

Theorem 3. *There is an $\mathcal{O}(\ell n^2 + n^3)$-time algorithm for Temporal Path Cover on temporal oriented trees with n vertices and at most ℓ many labels per arc. Furthermore, temporal oriented trees satisfy the Dilworth property.*

For the rest of this section, $\mathcal{T} = (T, \lambda)$ shall denote a temporal oriented tree with n vertices and at most ℓ-many labels per edge. We construct the *connectivity graph of* $\mathcal{T}$, denoted by G, as follows: $V(G) = V(T)$ and $E(G) = \{uv \mid u \neq v \text{ and } u \text{ and } v \text{ are temporally connected}\}$. In other words, the connectivity graph of a temporal oriented tree connects vertices that are temporally connected. Observe that G can be constructed in $\mathcal{O}(\ell n^2)$-time. The next observation follows immediately from the definition.

Observation 10. *A set S of vertices of $\mathcal{T}$ is a temporal antichain if and only if S induces an independent set in G.*

We have the following relationship between temporal paths in $\mathcal{T}$ and cliques in G.

Lemma 11. *Let S be a set of vertices of $\mathcal{T}$. Then S is contained in a temporal path in $\mathcal{T}$ if and only if S is contained in a clique of G.*

Proof. Let S be contained in temporal path P in $\mathcal{T}$. Let $u_1, u_2, \ldots, u_k$ where $k = |S|$, be the ordering of the vertices in S as they are encountered while traversing P from the source to the sink. Notice that, for each $1 \leq i < j \leq k$, there is a temporal path from u_i to u_j. Therefore, u_i is adjacent to u_j in G. Hence, S is contained in a clique of G.

Let S be contained in a clique of G and S' be a maximal complete subgraph of G such that $S \subseteq V(S')$. Now, we orient the edges of S' to create a digraph $\overrightarrow{S'}$ as follows. For an edge $uv \in E(S')$, we introduce an arc $\overrightarrow{uv} \in A(\overrightarrow{S'})$ if there is a temporal path from u to v in $\mathcal{T}$. Since $\mathcal{T}$ is acyclic, $\overrightarrow{S'}$ is a transitive tournament. Hence, there is an ordering $u_1, u_2, \ldots, u_k$ of the vertices of S' where $k = |V(S')|$ such that for $1 \leq i < j \leq k$, there is a temporal path from u_i to u_j in $\mathcal{T}$. Now, consider any temporal path P from u_1 to u_k in $\mathcal{T}$. (P exists as $\overrightarrow{u_1 u_k} \in A(\overrightarrow{S'})$. Since $\mathcal{T}$ is a temporal oriented tree, P will contain all vertices of S' and therefore of S. □

Following is an immediate corollary of the above.

Corollary 12. *The minimum cardinality of a temporal path cover of $\mathcal{T}$ is equal to the minimum cardinality of a clique cover of G.*

We will often use the following lemma about vertex-intersecting temporal paths between pairs of vertices.

Lemma 13. *Let $\{u, v, w, x\} \subseteq V(T)$ be four vertices such that any temporal path from u to v has a vertex in common with any temporal path from w to x. Then, there is a temporal path from u to x, or a temporal path from w to v.*

Proof. Assume that there is no temporal path from u to x. Let y be the vertex of a temporal path from w to x that is closest to u in T. Let t be the smallest integer such that there is a temporal path from u to v that reaches y at time-step t. Observe that no temporal path from y to x can start at time-step $t' > t$ since, otherwise, there would be a temporal path from u to x. This implies that all temporal paths between w and x reach y at time-step $t'' \leq t$. Let P_1 be a temporal path from w to y which is also a subpath of a temporal path from w to x. Let P_2 be a temporal path from y to v which is also a subpath of a temporal path from u to v. The above arguments imply that the arc incident with y in P_1 has time-step at most t. Similarly, the arc incident with y in P_2 has time-step strictly greater than t. Hence, the concatenation of P_1 and P_2 is a temporal path in $\mathcal{T}$ from w to v. □

4.1 The case of holes

In this subsection, we will show that the connectivity graph G does not contain any holes. We use the following lemma.

Lemma 14. *Let H be an induced cycle of length at least 4 in G. Then, for every vertex $v \in V(H)$ and every arc $\overrightarrow{a}$ of T incident with v, the vertices of $H \setminus \{v\}$ lie in the same connected component of $T \setminus \{\overrightarrow{a}\}$.*

Proof. For the sake of contradiction, let there exist vertices $\{u, v, w\} \subseteq V(H)$ and an arc $\overrightarrow{a}$ of T incident with v such that u and w lie in two different connected components of $T' = T \setminus \{\overrightarrow{a}\}$. Let C_u and C_w be the sets of vertices of $H \setminus \{v\}$ contained in the same connected component as u and w, respectively. Since $H \setminus \{v\}$ is connected, there exist $u' \in C_u$ and $w' \in C_w$ such that $u'w' \in E(H)$ *i.e.* $u'w' \in E(G)$. Hence, there is a temporal path P from u' to w' or w' to u' in $\mathcal{T}$. Since T is a tree, P must contain v. Lemma 11 implies that $\{u', v, w'\}$ forms a subset of a clique in G, and therefore $\{u', v, w'\}$ forms a triangle. But this contradicts that H is a hole. □

Going forward, we need the following notations. For an edge $e = uv \in E(G)$, let Q_e denote a temporal path from u to v or v to u in $\mathcal{T}$. For an induced cycle H of length at least 4 in G, let T_H denote the smallest connected subtree of T containing all vertices of H. Lemma 14 implies that every vertex of H must be a leaf in T_H. For a vertex $v \in V(H)$, let $\overrightarrow{a}(v)$ be the arc incident with v in T_H. Let H be an induced cycle of length at least 4 in G. We can partition the vertex set of H into two sets $IN(H)$ and $OUT(H)$ as follows: a vertex $v \in V(H)$ is in $IN(H)$ if $\overrightarrow{a}(v)$ is directed towards v, and otherwise v is in $OUT(H)$.

For a vertex $v \in IN(H)$, notice that both neighbors of v in H must lie in $OUT(H)$, and vice versa, since they must be connected by a directed path in T. Hence, H is bipartite, and therefore G does not contain any odd hole (*i.e.*, a hole with an odd number of vertices):

Lemma 15. *The connectivity graph G does not contain any odd hole.*

Without loss of generality, we assume in the following that $OUT(H)$ (resp. $IN(H)$) contains every odd-indexed (resp. even-indexed) vertex of H. For an even hole H whose vertices are cyclically ordered as $u_1, u_2, \ldots, u_k$, we use a cyclic definition of addition, so $k + 1 = 1$. We first prove the following lemmas.

Lemma 16. *Let H be an even hole in the connectivity graph G. Then, for every i, $Q_{u_i u_{i+1}}$ and $Q_{u_{i+2} u_{i+3}}$ share a common vertex.*

Proof. Assume by contradiction that $Q_{u_iu_{i+1}}$ and $Q_{u_{i+2}u_{i+3}}$ are vertex-disjoint. Assume without loss of generality that $Q_{u_iu_{i+1}}$ goes from u_i to u_{i+1}. Note that, since each vertex of the hole is a leaf of T_H as a consequence of Lemma 14, the two paths $Q_{u_iu_{i+1}}$ and $Q_{u_{i+1}u_{i+2}}$ have to share a common vertex other than u_{i+1} (its neighbour in T_H). By the same reasoning, $Q_{u_{i+1}u_{i+2}}$ and $Q_{u_{i+2}u_{i+3}}$ share a common vertex other than u_{i+2}. Hence, since the three paths $Q_{u_iu_{i+1}}$, $Q_{u_{i+1}u_{i+2}}$ and $Q_{u_{i+2}u_{i+3}}$ are in T_H, and $Q_{u_iu_{i+1}}$ and $Q_{u_{i+2}u_{i+3}}$ are vertex-disjoint, there is an arc $\overrightarrow{a}$ contained in $Q_{u_{i+1}u_{i+2}}$ that separates $Q_{u_iu_{i+1}}$ and $Q_{u_{i+2}u_{i+3}}$.

Removing $\overrightarrow{a}$ from T partitions the vertices of H into two sets H_1 and H_2: H_1 (resp. H_2) contains the vertices of H that are in the same part of $T \setminus \overrightarrow{a}$ as u_{i+1} (resp. u_{i+2}). Now, since H is a cycle, there is an edge u_ju_{j+1} such that (without loss of generality) $u_j \in H_1$, $u_{j+1} \in H_2$ and $(j, j+1) \neq (i+1, i+2)$. This implies that the path $Q_{u_ju_{j+1}}$ has to use $\overrightarrow{a}$ in $\mathcal{T}$, and thus $Q_{u_{i+1}u_{i+2}}$ and $Q_{u_ju_{j+1}}$ share a common vertex. Hence, Lemma 13 implies that there is a temporal path from u_{j+1} to u_{i+1} or from u_{i+2} to u_j. However, since $j \neq i+3$ ($u_j \in H_1$ and $u_{i+3} \in H_2$) and $j+1 \neq i$ ($u_{j+1} \in H_2$ and $u_i \in H_1$), both temporal paths would induce a chord in H, a contradiction. □

Lemma 17. *The connectivity graph G does not contain any hole of size 6.*

Proof. Assume by contradiction that there is a hole on six vertices $u_1, \ldots, u_6$. We know that $Q_{u_1u_2}$ and $Q_{u_4u_5}$ are vertex-disjoint (since otherwise, by Lemma 13, at least one of the chords u_1u_4 or u_2u_5 would exist). The u_i's are leaves of T_H, so $Q_{u_1u_2}$ and $Q_{u_1u_6}$, being paths with a common leaf in the same subtree, share at least one common vertex other than u_1 (its neighbour in T_H), let v be the last (with respect to the orientation of T) vertex in their common subpath. Now, $Q_{u_5u_6}$ has a common vertex with both $Q_{u_1u_2}$ (by Lemma 16) and $Q_{u_1u_6}$ (the neighbour of u_6 in T_H), so it has to contain v by the Helly property of subtrees of a tree. By the same reasoning, $Q_{u_4u_5}$ and $Q_{u_5u_6}$ share at least one common vertex other than u_5 (its neighbour in T_H), let w be the last vertex in their common subpath. The Helly property of subtrees of a tree again implies that both $Q_{u_2u_3}$ and $Q_{u_3u_4}$ have to contain w, since they pairwise intersect with $Q_{u_4u_5}$. But this means that $Q_{u_2u_3}$ and $Q_{u_5u_6}$ share both v and w as common vertices, and so by Lemma 13 there is at least one of the two chords u_2u_5 or u_3u_6, a contradiction. □

We can now prove that there is no even hole in G:

Lemma 18. *The connectivity graph G does not contain any even hole.*

Proof. Assume by contradiction that G contains an even hole H on $k \geq 8$ vertices ($k = 6$ is impossible by Lemma 17). We know by Lemma 16 that both $Q_{u_3u_4}$ and $Q_{u_{k-1}u_k}$ intersect $Q_{u_1u_2}$, but do not intersect each other (otherwise, by Lemma 13, at least one of the edges u_3u_k or u_4u_{k-1} would exist, and both would be chords since $k \geq 8$), so there is an arc $\overrightarrow{a}$ in T that separates them. Removing $\overrightarrow{a}$ from T partitions the vertices of H into two sets H_1 and H_2: H_1 (resp. H_2) contains the vertices of H that are in the same part of $T \setminus \overrightarrow{a}$ as u_3 (resp. u_k). Now, since H is a cycle, there is an edge u_ju_{j+1} such that (without loss of generality) $u_j \in H_1$ and $u_{j+1} \in H_2$. This implies that the path $Q_{u_ju_{j+1}}$ has to use $\overrightarrow{a}$ in $\mathcal{T}$, and thus $Q_{u_1u_2}$ and $Q_{u_ju_{j+1}}$, both containing $\overrightarrow{a}$, share a common vertex. Hence, Lemma 13 implies that there is a temporal path from u_{j+1} to u_2 or from u_1 to u_j. However, since $j \neq k$ ($u_j \in H_1$ and $u_k \in H_2$) and $j+1 \neq 3$ ($u_{j+1} \in H_2$ and $u_3 \in H_1$), by Lemma 13 both temporal paths would induce a chord in H, a contradiction. □

4.2 The case of anti-holes

In this subsection, we will show that the connectivity graph G does not contain any anti-hole. For an anti-hole H, let its vertices be circularly ordered as $u_1, u_2, \ldots, u_k$ as they are encountered while traversing the complement of H (which is a hole). Let $ODD(H)$ (resp. $EVEN(H)$) denote the set of vertices with odd (resp. even) indices.

Lemma 19. *The connectivity graph G does not contain any anti-hole.*

Proof. Assume by contradiction that G contains an anti-hole H with k vertices. If $k = 5$, then H is a hole, which contradicts Lemma 15; hence, assume $k \geq 6$.

When k is odd, let $F_1 = ODD(H) \setminus \{u_k\}, F_2 = EVEN(H)$. When k is even, let $F_1 = ODD(H), F_2 = EVEN(H)$. Observe that $|F_1| = |F_2| \geq 3$ and both sets induce (vertex-disjoint) cliques in G. By Lemma 11, there are temporal paths P_1 and P_2 in $\mathcal{T}$ containing F_1 and F_2, respectively, which we can assume are minimal vertex-inclusion-wise (so that, for each $i \in \{1,2\}$, both end-vertices of P_i lie in F_i). For $i \in \{1,2\}$, let v_i and w_i denote the source and sink of P_i, respectively. We have two cases.

Case 1: $V(P_1) \cap V(P_2) = \emptyset$**.** Let Q be the shortest temporal path that contains vertices from both P_1 and P_2. Let p_1, p_2 be the end-vertices of Q that lie on P_1 and P_2, respectively. Since for each $i \in \{1,2\}$ and $Z \in \{F_1, F_2\}$, $N_G(w_i) \cap Z \neq \emptyset$, Q is oriented from p_1 to p_2, or vice versa. Without loss of generality, assume that Q is oriented from p_2 to p_1. Then, necessarily $p_2 = w_2$, since otherwise w_2 is not temporally connected with any vertex of F_1, a contradiction. By a similar argument, we have $p_1 = v_1$. Now, consider the clique induced by $N(v_2) \cap F_1$. Due to Lemma 11, all vertices of $N(v_2) \cap F_1$ and v_2 itself are contained in a temporal path, which also necessarily contains w_2. Hence all of F_2 (P_2, even) is in a temporal path containing v_1, since the path has to go through v_1 to reach other vertices of F_1, and so $F_2 \cup \{v_1\}$ forms a clique. This is a contradiction as v_1 necessarily has at least one non-neighbor in F_2.

Case 2: $V(P_1) \cap V(P_2) \neq \emptyset$**.** Let Q denote the maximal vertex-inclusion-wise path that is common to both P_1 and P_2, *i.e.*, the path induced by the set $V(P_1) \cap V(P_2)$. Note that Q does not contain any vertex from H, since a vertex of H in Q would be temporally connected to every other vertex of $F_1 \cup F_2$, a contradiction. Let p denote source of Q and for each $i \in \{1,2\}$ let Q_i (resp. Q'_i) be the subpath of P_i between p and w_i (resp. p and v_i).

Note that no vertex of $Q'_1 \setminus \{p\}$ can be in a directed path with any vertex of $Q'_2 \setminus \{p\}$. Similarly, no vertex of $Q_1 \setminus \{p\}$ can be in a directed path with any vertex of $Q_2 \setminus \{p\}$. Thus, the two subgraphs of the connectivity graph G induced by the vertices of $(V(Q_1) \cup V(Q_2)) \setminus V(Q)$ and $(V(Q'_1) \cup V(Q'_2)) \setminus \{p\}$ each induce the complement of a complete bipartite graph. As H does not contain any complement of a 4-cycle as an induced subgraph, this implies that there are exactly three vertices of H in each of these two subsets of vertices (since Q does not contain any vertex of H). In particular, H has size either 6 or 7.

Without loss of generality, we assume that Q'_1 contains only one vertex of H, which must be v_1. Thus, there are two vertices of H in Q'_2: v_2 and another vertex, say, v'_2. Since F_1 and F_2 both have size 3, the vertices of H in Q_1 are w_1 and (say) w'_1, and the only vertex of H in Q_2 is w_2. Now, observe that if v_2 is contained in a temporal path with w_1, then v_2, v'_2, w'_1 and w_1 are in a common temporal path. This is not possible, since in H, there is either one or two non-edges among these four vertices (depending on whether H has size 7 or 6). Thus, w_1 and v_2 are in no common temporal path. Since v_2 has no non-neighbour in H other than v_1 and w_1, v_2 and w'_1 are in a common temporal path, that also contains v'_2. Thus, $\{v_2, v'_2, w'_1\}$ form a clique in H. Similarly, $\{v'_2, w'_1, w_1\}$ also form a clique in H. If H had size 6, v'_2 and w'_1 would need to be non-neighbours in H (since w_1 already has two non-neighbours in H), a contradiction. Thus, H has size 7, and the two non-neighbours in H of u_7 (the vertex of H not in $F_1 \cup F_2$) are v'_2 and w'_1 (since they are the only ones without two non-neighbours in H). But u_7 has to be temporally connected to all of v_1, v_2, w_1 and w_2, so u_7 has to be in Q. But any temporal path from v_2 to a vertex of Q has to contain v'_2, and so u_7 and v'_2 are temporally connected, a contradiction. This completes the proof. □

4.3 Completion of the proof of Theorem 3

Lemmas 15, 18 and 19 imply that the connectivity graph of a temporal oriented tree is weakly chordal. Note that this cannot be strengthened to chordal, as there are temporal oriented trees whose connectivity graphs contain induced 4-cycles: let $\lambda(\overrightarrow{s_1c}) = \lambda(\overrightarrow{s_2c}) = 1$ and $\lambda(\overrightarrow{ct_1}) = \lambda(\overrightarrow{ct_2}) = 2$, the vertices s_1, t_1, s_2 and t_2 induce a C_4 in the connectivity graph. Corollary 12 implies the correspondence between a minimum temporal path cover of a temporal oriented tree and a minimum clique cover of the corresponding connectivity graph. We then conclude using Theorem 8 for the algorithm. Observation 10, Corollary 12 and Theorem 8 together give the Dilworth property.

5 Temporally Disjoint Path Cover on temporal oriented trees

Theorem 4. *Temporally Disjoint Path Cover is NP-hard on temporal oriented trees.*

Proof. The reduction is inspired from Theorem 1 in [29, 30]. However, in [29, 30], the terminal vertices of the paths are fixed, which is not the case in our problem. Thus, nontrivial additions are needed. We reduce from Unary Bin Packing, which is NP-complete [17].

> Unary Bin Packing
> Instance: A list of item sizes $(x_1, \dots, x_n)$, a number of bins b, each of size B. $x_1, \dots, x_n, b, B$ are integers encoded in unary, and verify $\sum_{i=1}^{n} x_i = bB$.
> Question: Is it possible to assign every item to a bin, filling all the bins?

The idea of the reduction will be to have pairs of vertices serving as bins, each with B leaves, and to have vertices representing, for each item, the bins that are unused by this item.

We construct the following temporal oriented tree $\mathcal{T} = (T, \lambda)$:

- $V(T) = \{c\} \cup \bigcup_{i=1}^{b} \left\{ \{s_i, t_i\} \cup \bigcup_{j=1}^{B} \{r_i^j, u_i^j\} \right\} \cup \bigcup_{i=1}^{n} \bigcup_{j=1}^{(x_i-1)(b-1)} \{v_i^j, w_i^j\}$
- $A(T) = \bigcup_{i=1}^{b} \left\{ \{\overrightarrow{s_i c}, \overrightarrow{c t_i}\} \cup \bigcup_{j=1}^{B} \{\overrightarrow{r_i^j s_i}, \overrightarrow{t_i u_i^j}\} \right\} \cup \bigcup_{i=1}^{n} \bigcup_{j=1}^{(x_i-1)(b-1)} \{\overrightarrow{v_i^j c}, \overrightarrow{c w_i^j}\}$

For the sake of simplicity, we will use *layers* to represent the time labels: for each item i, the layer λ_i will assign to every arc a subset of $\{1, \dots, 2bx_i+4\}$. Thus, for an arc a, we have $\lambda(a) = \bigcup_{i=1}^{n} \{\ell + \sum_{j=1}^{i-1}(2bx_i+4) \mid \ell \in \lambda_i(a)\}$. This allows us to describe the layers starting with label 1. We call two time labels *2-successive* if they differ by 2. The time labels in layer i are as follows.

- For every $j \in \{1, \dots, b\}$ and $k \in \{1, \dots, B\}$: $\lambda_i(\overrightarrow{r_j^k s_j}) = \{$every 2-successive label between $2(j-1)x_i + 1$ and $2jx_i - 1\}$; $\lambda_i(\overrightarrow{s_j c}) = \{$every 2-successive label between $2(j-1)x_i + 2$ and $2jx_i\}$; $\lambda_i(\overrightarrow{c t_j}) = \{$every 2-successive label between $2(j-1)x_i + 3$ and $2jx_i + 1\}$; $\lambda_i(\overrightarrow{t_j u_j^k}) = \{$every 2-successive label between $2(j-1)x_i + 4$ and $2jx_i + 2\}$.
- For every $j \in \{1, \dots, (x_i-1)(b-1)\}$: $\lambda_i(\overrightarrow{v_i^j c}) = \bigcup_{k=1}^{b} \{$the $x_i - 1$ first labels of $\overrightarrow{c t_k}\}$; $\lambda_i(\overrightarrow{c w_i^j}) = \bigcup_{k=1}^{b} \{$the $x_i - 1$ highest labels of $\overrightarrow{s_k c}\}$. For a given k, those are called the *bin-k-labels*.

A layer of this construction is depicted in Figure 5. The number of vertices is $1 + 2b + 2bB + \sum_{i=1}^{n}(x_i - 1)(b-1) = 2b(bB - n + 1) + 2n + 1$, among which $2b(bB - n) + 2n$ are leaves (half of them are sources, the other half sinks). The number of different time labels is $2bx_i + 4$ for layer i, and thus a total of $2b(bB + 4n)$. Hence, the reduction is polynomial.

We now prove that there is a valid assignment of every item to a bin if and only if there is a temporally disjoint path cover of $\mathcal{T}$ of size $b(bB - n) + n$.

($\Rightarrow$) Suppose that $f : \{1, \dots, n\} \to \{1, \dots, b\}$ is a valid assignment (so $\sum_{i \in f^{-1}(j)} x_i = B$ for every j). In every layer i, we take x_i (r, u)-*paths* $(\overrightarrow{r_{f(i)}^j s_{f(i)}}, 2(f(i)-1)x_i + 2a), (\overrightarrow{s_{f(i)} c}, 2(f(i)-1)x_i + 2a + 1), (\overrightarrow{c t_{f(i)}}, 2(f(i)-1)x_i + 2a + 2), (\overrightarrow{t_{f(i)} u_{f(i)}^k}, 2(f(i)-1)x_i + 2a + 3)$ using uncovered $r_{f(i)}^j$'s and $u_{f(i)}^k$'s and for $a \in \{1, \dots, x_i\}$; as well as all the paths from v_i^j to w_i^k such that the arc $\overrightarrow{v_i^j c}$ is used with a bin-k-label for $k \neq f(i)$. The first paths will always be possible, and will cover every r_i^j and u_i^j once all the layers are done, since we will use exactly B paths for every i. The other paths can also clearly be constructed,

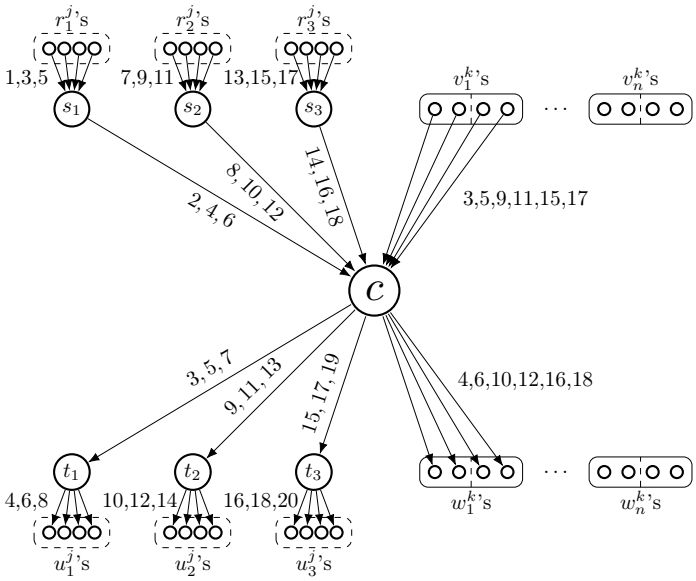


Figure 5: Layer 1 of the reduction of the proof of Theorem 4, with $x_1 = 3$, $b = 3$, $B = 4$. The only v_j^k's and w_j^k's linked with c in this layer are those with $j = 1$.

since whenever $k \neq f(i)$ the bin-k-labels are not used by the first paths, and so we will cover all the v_i^j's and w_i^j's. Hence, we obtain a temporally disjoint path cover of $\mathcal{T}$ (c and the s_i's and t_i's are clearly covered) of size $\sum_{i=1}^{n}(x_i + (x_i - 1)(b-1)) = b(bB - n) + n$.

($\Leftarrow$) Suppose that there is a temporal path cover of $\mathcal{T}$ of size $b(bB - n) + n$. The cover is of size twice the number of leaves, so each path in the cover will contain two leaves. Since every path between two leaves has to go through c and the paths are temporally disjoint, there can be at most bx_i paths in layer i, with equality if and only if all in-arcs with successive labels from 2 to $2bx_i$ are used.

The vertices v_i^j's are linked with c only in layer i, so they are covered in layer i by an arc $\overrightarrow{v_i^j c}$ at time d (note that d is odd); sort them in a sequence S ordered by the d's. We call a subsequence of S *2-successive* if the d's are 2-successive.

Claim 1. *Each 2-successive subsequence of length k prevents $k+1$ (r,u)-paths from going through c.*

Proof. Each 2-successive subsequence $S' = (d, d+2, \ldots, d+2k-2)$ (recall that d is odd) induces paths that occupy c at all times between d and $d+2k-1$ (since the last path needs to leave c in order to allow another path to occupy c in turn); but due to the difference of parity between the time labels of the arcs $\overrightarrow{u_i^j c}$ and $\overrightarrow{s_k c}$, S' will prevent at least $|S'| + 1 = k + 1$ (r,u)-paths from going through c, since no such path will be able to go through c in the interval between $d-1$ and $d+2k-1$. □

Note that having a path start in one layer and end in another layer will lower the maximum number of paths restricted to their layers by 1 for each of both, while gaining at most one path with arcs in two different layers. Furthermore, while the theoretical maximum number of paths in layer i is bx_i, the v_i^j's are covered in layer i, and thus by Claim 1 there can be at most $bx_i - (b-1)$ paths in layer i. Since the number of paths in the cover is $b(bB-n)+n = \sum_{i=1}^{n}(bx_i - (b-1))$, there is no path having arcs in two different layers. Similarly, the only paths in the cover are (r,u)-paths and (v,w)-paths (since, otherwise, there would be even fewer paths in the layer).

Claim 2. *There are exactly x_i (r,u)-paths in layer i.*

Proof. Due to the definition of λ_i, there are at least $b-1$ 2-successive subsequences, each of length at most $x_i - 1$. By Claim 1, they prevent at least $(b-1)(x_i-1)+(b-1) = (b-1)x_i$ (r,u)-paths from going through c in layer i (since the number of (v,w)-paths is fixed, increasing the number of 2-successive subsequences will decrease their sizes, but will end up increasing the number of time labels during which c is occupied). Since there are at most bx_i paths per layer, there are at most x_i (r,u)-paths in layer i, with equality if and only if there are exactly $b-1$ 2-successive subsequences. As this holds for every layer, the path cover is of size $b(bB-n)+n$, and $(b-1)(bB-n)$ paths are necessary to cover the v_j^k's and w_j^k's, there has to be bB (r,u)-paths in all the layers; thus there are exactly x_i such paths in layer i. □

Claim 3. *All the (r,u)-paths of a layer i have to all go through the same s_j.*

Proof. Assume by contradiction that there are (r,u)-paths in the same layer using s_j and s_k for $j \neq k$. Then, in order to cover the v_i's and w_i's, we need to have at least b 2-successive subsequences (at least 1 for each of j and k, and at least $b-2$ for the other bins). By Claim 1 and as in the proof of Claim 2, this implies that there will be less than x_i (r,u)-paths in the layer, which contradicts Claim 2, so all the (r,u)-paths go through the same s_j.

The same argument can be applied to show that all (r,u)-paths go through the same $t_{j'}$, and that $j = j'$. □

Hence, we can construct the item assignment function f as follows: for every item i, let j be the integer such that there are x_i (r,u)-paths in layer i going through s_j and t_j; we define $f(i) = j$. Claim 3 implies that f will assign each item to exactly one bin. Moreover, by our construction, there are exactly B (r,u)-paths going through each s_j, x_i of which at layer i for $i \in f^{-1}(j)$ by Claim 2, and thus it is a correct assignment: $\sum_{i \in f^{-1}(j)} x_i = B$ for each bin j. □

We also show the following.

Proposition 20. *There are temporal oriented trees (whose underlying digraph is a star) that do not satisfy the TD-Dilworth property.*

Proof. Consider the temporal oriented tree $\mathcal{S}_k = (S_k, \lambda)$, with $V(S_k) = \{s_1, \dots, s_k\} \cup \{t_1, \dots, t_k\} \cup \{c\}$, $A(S_k) = \bigcup_{i=1}^{k} \{\overrightarrow{s_i u}, \overrightarrow{u t_i}\}$, and $\lambda(\overrightarrow{s_i u}) = 1$ and $\lambda(\overrightarrow{c t_i}) = 2$ for $i \in \{1, \dots, k\}$. Now, as depicted on Figure 6, the s_i's (or the t_i's) form a (maximum-size) temporal antichain of size k. Since c is a cut-vertex with all its in-arcs having the same time label, there can only be one path using c to cover both an s_i and a t_i, and thus every other vertex has to be covered individually. Hence, we need at least $2k-1$ temporal paths in any temporal path cover of $\mathcal{S}_k$ (and it is easy to construct such a cover). □

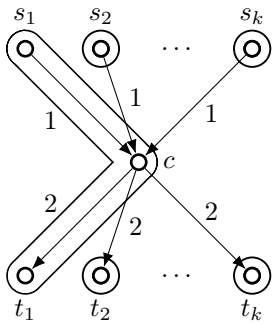


Figure 6: $\mathcal{S}_k$, a temporal oriented tree with a maximum-size temporal antichain of size k and a minimum-size TD-PC of size $2k-1$.

6 Subclasses of temporal oriented trees

Theorem 5. *Temporal Path Cover and Temporally Disjoint Path Cover can be solved in time:*
(a) $\mathcal{O}(\ell n)$ on temporal oriented lines;
(b) $\mathcal{O}(\ell n^2)$ on temporal rooted directed trees;
where ℓ is the maximum number of labels per arc and n is the number of vertices. Furthermore, both classes satisfy the TD-Dilworth property.

Proof. (a) Let $\mathcal{P} = (P, \lambda)$ be a temporal oriented line, and let v be a leaf of P. We construct $\mathcal{C}$ as follows. Assume that v is incident with an in-arc $\overrightarrow{uv}$. We construct a maximum-length temporal path that covers v. Set $(b, c) = (u, v)$, $\ell = \max \lambda(\overrightarrow{uv})$, and apply the following routine: while b is incident with an in-arc $\overrightarrow{ab}$, if there is a time label smaller than ℓ in $\lambda(\overrightarrow{ab})$, add $\overrightarrow{ab}$ to the path, update $(b, c) = (a, b)$ and $\ell = \max\{k \in \lambda(\overrightarrow{ab}) \mid k < \ell\}$. When the routine stops, add the path to $\mathcal{C}$, remove its vertices from P, and start again on a new leaf (or return $\mathcal{C}$ if P is empty). If v was incident with an out-arc, we would do the same but with out-arcs, start with the smallest possible time label, and update $\ell = \min\{k \in \lambda(\overrightarrow{ab}) \mid k > \ell\}$.

This algorithm computes its output in time $\mathcal{O}(\ell n)$: every arc is visited at most once, but we need to parse the time labels in order to see whether we can keep on extending the path or not.Furthermore, the set of leaves v where we start the routine are a temporal antichain: assume on the contrary that v_1 and v_2 are such vertices that are temporally connected, and assume without loss of generality that there is a path from v_1 to v_2 in the underlying oriented path; in this case, our algorithm would have added v_1 to the path that started being computed at v_2, a contradiction. Hence, $\mathcal{C}$ is a temporally disjoint path cover with the same size as a temporal antichain, proving that it is minimum-size and that temporal oriented lines satisfy the TD-Dilworth property.

(b) We give an algorithm that solves Temporal Path Cover on a temporal rooted directed tree $\mathcal{T} = (T, \lambda)$. First, we sort the vertices of T with respect to their topological distance from the root in T (with the highest distances first). Then, we construct a maximum-length temporal path covering the first uncovered vertex (which will be a sink of that path), until $\mathcal{T}$ is fully covered.

Note that this algorithm outputs $\mathcal{C}$ which is clearly is a temporal path cover: every vertex is covered by some path of $\mathcal{C}$. Furthermore, it is an adaptation of the algorithm for temporal oriented lines: instead of successive leaves, we construct the paths from successive leaves with highest topological distance from the root. We will show that $\mathcal{C}$ is minimum-size, and later we will explain how to modify the algorithm in order to obtain a minimum-size TD-PC.

Let S be the set of sinks of paths of $\mathcal{C}$. First, let v_i and v_j be two vertices of S (without loss of generality, assume that v_i was covered by the algorithm after v_j). They cannot be temporally connected, since otherwise, the graph being a temporal rooted directed tree, one of them is necessarily the predecessor of the other in a path from the root, and thus the maximum-length temporal path ending in v_j would necessarily contain v_i, since there is a temporal path from v_i to v_j, and thus v_i would have been covered at this step and cannot be in S. Hence, S is a temporal antichain.

We now prove that S is maximum-size. Assume by contradiction that there is a temporal antichain S' with $|S'| > |S|$. However, by definition, no two vertices in S' can be covered by the same temporal path of $\mathcal{C}$, and thus $|S'| = |\mathcal{C}|$. But $|\mathcal{C}| = |S|$, a contradiction. Thus, S is a maximum-size temporal antichain of the temporal rooted directed tree. Since the temporal antichain number is a lower bound for the temporal path cover number, this implies that the temporal path cover $\mathcal{C}$ that the algorithm constructed is minimum-size, and thus that temporal rooted directed trees satisfy the Dilworth property.

We now modify the algorithm to obtain a minimum-size temporally disjoint path cover. Indeed, we can see that the maximum-length temporal path construction, which is executed for every vertex of S, can re-cover some vertices that had already been covered at a previous step. Let v_i and v_j be two vertices of S such that their maximum-length temporal paths constructed by the algorithm P_i and P_j intersect. Since the graph is a temporal rooted directed tree, we can divide P_i and P_j into the following parts, without loss of generality: $P_i = P_i^{\text{top}} \cup (P_i \cap P_j) \cup P_i^{\text{bot}}$ and $P_j = (P_i \cap P_j) \cup P_j^{\text{bot}}$, where $P_i^{\text{top}} \cap P_j = P_i^{\text{bot}} \cap P_j = P_j^{\text{bot}} \cap P_i = \emptyset$ (note that we can have $P_i^{\text{top}} = \emptyset$). Hence, we can modify the algorithm by adding a loop that, for each such pair (P_i, P_j), defines those subpaths and then removes $P_i \cap P_j$ from P_j. Now, $\mathcal{C}$ will still be a temporal path cover, but

the paths will be vertex-disjoint and thus temporally disjoint, and its size will not change. This implies that temporal rooted directed trees satisfy the TD-Dilworth property (contrasting the general temporal oriented trees), and thus the modified algorithm outputs the optimal solution for those two problems. The result of the algorithm and its modification is depicted in Figure 7.

Finally, one can check that the algorithm and its modification compute $\mathcal{C}$ in time $\mathcal{O}(\ell n^2)$. For each vertex in the antichain S, we have to construct the maximum-length temporal path. This can be done in time $\mathcal{O}(\ell n)$ by taking at every arc the largest label that allows to extend the path, thus we have to parse all the labels of every arc along the path, which can be of linear-size in the worst case. Since we can have a linear number of antichain vertices, we have a complexity of $\mathcal{O}(\ell n^2)$ to get the temporal path cover. The modification to make it temporally disjoint can be done in $\mathcal{O}(n^2)$ time afterwards. □

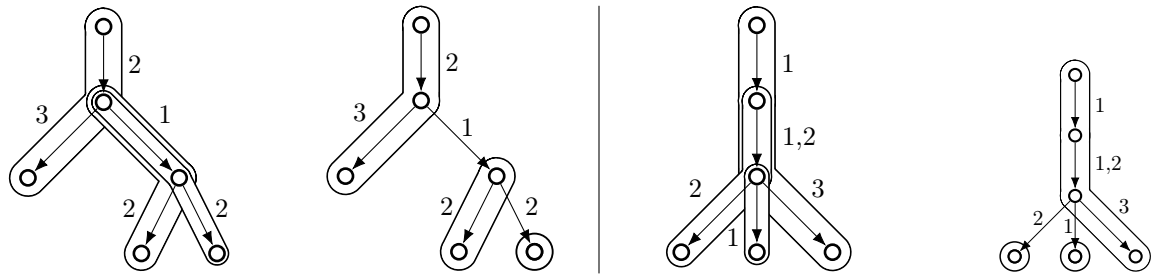


Figure 7: Minimum-size temporal path covers and temporally disjoint path covers of the same temporal rooted directed tree (on the left, with one label per arc; on the right, with any labels per arc), as computed by our algorithm and its modification in the proof of Theorem 5.

7 Algorithms for temporal digraphs of bounded treewidth

Recall that an algorithm is FPT with respect to some parameter k of the input, if it runs in time $f(k)n^{\mathcal{O}(1)}$ for inputs of size n, where f is any computable function; the algorithm is XP for this parameter if the running time is in $n^{f(k)}$ [11]. We prove the following theorem.

Theorem 6. *There is an algorithm for* Temporally Disjoint Path Cover *on general temporal digraphs that is FPT with respect to the treewidth of the underlying undirected graph and the maximum number of labels per arc. For* Temporal Path Cover *on general temporal digraphs, there is an XP algorithm for the same parameter.*

To prove the theorem, we use the well-known concept of *nice tree decompositions* [27], which gives a very structured decomposition of a graph.

Definition 21. *A* nice tree decomposition *of an undirected graph $G = (V, E)$ is a rooted tree* $\mathtt{T}$ *where each node v is associated to a subset X_v of V called* bag*, and each internal node has one or two children, with the following properties.*

1. *The set of nodes of* $\mathtt{T}$ *containing a given vertex of G forms a nonempty connected subtree of* $\mathtt{T}$.
2. *Any two adjacent vertices of G appear in a common node of* $\mathtt{T}$.
3. *Each node of* $\mathtt{T}$ *belongs to one of the following types:* introduce*,* forget*,* join *or* leaf.
4. *A join node v has two children v_1 and v_2 such that $X_v = X_{v_1} = X_{v_2}$.*
5. *An introduce node v has one child v_1 such that $X_v \setminus \{x\} = X_{v_1}$, where $x \in X_v$.*
6. *A forget node v has one son v_1 such that $X_v = X_{v_1} \setminus \{x\}$, where $x \in X_{v_1}$.*

7. *A leaf node v is a leaf of $\mathtt{T}$ with $X_v = \emptyset$.*

8. *The tree $\mathtt{T}$ is rooted at a leaf node r with $X_r = \emptyset$.*

It is known that for any undirected graph of treewidth tw with n vertices, a tree-decomposition of width at most $2\,\mathrm{tw}$ can be computed in time $2^{\mathcal{O}(\mathrm{tw})}n$ [28], and the obtained tree decomposition can be transformed into a nice tree-decomposition of the same width with $\mathcal{O}(\mathrm{tw}\, n)$ bags in time $\mathcal{O}(\mathrm{tw}^2\, n)$ [27].

For the remainder of this section, we shall work with a temporal digraph $\mathcal{D} = (V, A_1, \ldots, A_{t_{\max}})$, and a nice tree decomposition $\mathtt{T}$ of the underlying undirected graph of D. For a node v of $\mathtt{T}$, let $\mathtt{T}_v$ denote the subtree of $\mathtt{T}$ rooted at v, and let $\mathcal{D}_v$ denote the temporal digraph induced by the union of the bags of nodes of $\mathtt{T}_v$.

The main idea behind the algorithm is to perform a bottom-up dynamic programming algorithm over $\mathtt{T}$. We can bound the number of partial solutions that can intersect a given bag, partly because of the following.

Observation 22. *Let $\mathcal{C}$ be a temporally disjoint path cover of $\mathcal{D}$. Then any arc of $\mathcal{D}$ appears in at most $t_{\max}$ many paths of $\mathcal{C}$.*

Consider an arbitrary temporally disjoint path cover $\mathcal{C}$ of $\mathcal{D}_v$. Observation 22 implies that the number of temporally disjoint paths of $\mathcal{C}$ that contain at least one arc from the digraph induced by X_v is at most the number of arcs in this digraph, times the number of time-steps at which each of the arcs is active in $\mathcal{D}$. This is at most $p = \binom{\mathrm{tw}}{2} \cdot t_{\max}$.

Based on these, we create the following temporal multi-digraph. Let $\mathcal{D}'$ be a copy of $\mathcal{D}$. Now; for each arc a with time labels $\lambda(a) = L \subseteq [t_{\max}]$, introduce $|L|$ many new arcs $a^1, \ldots, a^{|L|}$, each with a distinct time label of L. Note that any temporally disjoint path cover of $\mathcal{D}$ can be transformed into a temporally disjoint path cover of $\mathcal{D}'$ whose temporal paths are pairwise edge-disjoint. Therefore, from now on, we will consider $\mathcal{D}'$ instead of $\mathcal{D}$.

We now describe the states of our dynamic programming algorithm. To do so, for a temporally disjoint path cover $\mathcal{C}$ of $\mathcal{D}$, its *type* τ with respect to a node v of $\mathtt{T}$ is determined by the following elements:

- a partition $\mathcal{Q} = Q_0, Q_1, \ldots, Q_t$ of the arcs of $\mathcal{D}'$ inside X_v, where each part Q_i corresponds to a temporal path $P(Q_i)$ of $\mathcal{C}$ (note that this path may form a set of disconnected sub-paths inside X_v), and where the part Q_0 is reserved for those arcs that do not belong to any path of $\mathcal{C}$;
- for each part Q_i of $\mathcal{Q}$, the subset V_i of vertices of X_v that belong to $P(Q_i)$ (those are the endpoints of arcs in Q_i, together with those vertices of $P(Q_i)$ that are not incident with any arc in Q_i);
- for each part Q_i of $\mathcal{Q}$, the order of the vertices of V_i inside $P(Q_i)$, where $P(Q_i)$ is ordered from lowest to largest time label;
- for each part Q_i of $\mathcal{Q}$, the set of vertices x in V_i with one or two arcs in $P(Q_i)$ from x to a vertex y not in X_v, together with the time labels of these arcs in $P(Q_i)$, and whether the arc connects x to a vertex in $\mathcal{D}_v$ or in $\mathcal{D} \setminus \mathcal{D}_v$.

The total number of different types of solutions with respect to any node v is at most $p^p \times 2^{\mathrm{tw}+1} \times (\mathrm{tw}+1)! \times 2^{\mathrm{tw}+2} \times t_{\max}^2$ which is $2^{\mathcal{O}(p \log p)}$. For a type τ with respect to node v, its *size* is the bit-length of its encoding. A type τ with respect to v is said to be *consistent* if for each part Q_i of $\mathcal{Q}$, there is a subset of vertices of V_i whose ordering together with the arcs of Q_i, form a valid temporal path (in particular, all labels of Q_i must be distinct). Moreover, the labels of required arcs from a vertex x of X_v to a vertex y outside of X_v, must correspond to an actual arc label for some arc in $\mathcal{D}$ connecting x to some vertex outside of X_v. Moreover, every vertex of X_v must belong to some set V_i. Whether a type τ with respect to v is consistent can be checked in time proportional to p and the size of τ. For a node v of $\mathtt{T}$ and a solution type τ with respect to v, let $\mathit{opt}(v, \tau)$ denote the minimum size of a solution for $\mathcal{D}_v$ that is of type τ with respect to v. The dynamic programming algorithm computes $\mathit{opt}(v, \tau)$ by traversing the nice tree-decomposition $\mathtt{T}$ bottom-up and computes, for each node v, all the values for $\mathit{opt}(v, \tau)$. The computation depends on whether the current node of $\mathtt{T}$ is a leaf, forget, introduce or join node.

Leaf node: There is nothing to do since for a leaf node v, $X_v = \emptyset$, so there is no partial solution with respect to v.

Forget node: Let v be a forget node that has a child node v' such that $X_v = X_{v'} \setminus \{x\}$. For each possible consistent solution type τ with respect to v, we check which (consistent) solution types τ' with respect to v' are compatible with τ. Whether τ and τ' are compatible (meaning that, roughly speaking, τ corresponds to τ' by removing x) can be computed in time proportional to p, the size of τ and that of τ'. Among those, we discard those where x is required to have an arc to a vertex of $\mathcal{D} \setminus \mathcal{D}_v$ in its solution path (since x will never appear again in the tree-decomposition). We let $opt(v, \tau)$ be the minimum value among all values $opt(v', \tau')$ with τ' one of the non-discarded types compatible with τ.

Introduce node: Let v be an introduce node that has a child node v' such that $X_v = X_{v'} \cup \{x\}$. For each possible consistent solution type τ with respect to v, we check which solution types with respect to v' are compatible with it. Here, this means that τ can be obtained from τ' either as a new solution path with a single vertex, or by adding x to one of the solution paths described by τ' (through an arc of the correct label as described in τ'). If x forms a single path in τ, we let $opt(v, \tau)$ be the minimum over $opt(v', \tau') + 1$, where τ' is compatible with τ; otherwise, we take the minimum over all $opt(v', \tau')$ for compatible τ'.

Join node: Let v be a join node with children v_1 and v_2 and $X_v = X_{v_1} = X_{v_2}$. For any three possible solution types τ, τ_1, τ_2 that are consistent with respect to v, v_1 and v_2, respectively, we check if they are compatible. For this, the partitions of the arcs of X_v have to agree, as well as the order of vertices inside each solution path. The other elements should also be compatible. For example, if in τ_1 there is a vertex $x \in X_v$ that is required to have a single neighbour in $\mathcal{D}_{v_1} \setminus X_v$ in its solution path, and the same holds for τ_2 and $\mathcal{D}_{v_2} \setminus X_v$, then the types are not compatible, since combining them would give two such neighbours to x in $\mathcal{D}_v \setminus X_v$. Similarly, if in τ_1, x is required to have a neighbour in $\mathcal{D}_{v_1} \setminus X_v$ in its solution path and another neighbour in $\mathcal{D} \setminus \mathcal{D}_{v_1}$, τ_1 is compatible with τ and τ_2 if in τ_2, x is required to have a neighbour in $\mathcal{D}_{v_2} \setminus X_v$ in its solution path and another neighbour in $\mathcal{D} \setminus \mathcal{D}_{v_2}$, but in τ, x is required to have two neighbours in $\mathcal{D}_v \setminus X_v$ in its solution path. Other similar cases arise as well. For three compatible types τ, τ_1, τ_2, $opt(v, \tau)$ is obtained from $opt(v_1, \tau_1) + opt(v_2, \tau_2)$ by subtracting the number of solution paths that intersect the bag (and that would otherwise be counted twice).

This dynamic programming algorithm for Temporally Disjoint Path Cover takes $2^{\mathcal{O}(p \log p)} n$ time, which is an FPT running time of $2^{\mathcal{O}(\mathrm{tw}^2 t_{\max} \log(\mathrm{tw}\, t_{\max}))} n$.

For Temporal Path Cover, the algorithm is similar, however, as the paths are not necessarily disjoint, the type of a solution with respect to v must also contain the information of how many times a given part of $\mathcal{Q}$, representing a solution path with a certain intersection with the subgraph induced by X_v, appears in the solution $\mathcal{C}$. Thus, the number of possible solution types becomes $k^{\mathcal{O}(p \log p)}$, where k is the solution size. We obtain a running time of $k^{\mathcal{O}(p \log p)}$, which is an XP running time of $n^{\mathcal{O}(\mathrm{tw}^2 t_{\max} \log(\mathrm{tw}\, t_{\max}))}$ since we can assume $k \leq n$.

8 Conclusion

We have initiated the study of two natural path covering problems in temporal DAGs, which, in the static case, are related to Dilworth's theorem and are polynomial-time solvable. Both problems become NP-hard for temporal DAGs, even in a very restricted setting. Interestingly, and somewhat unexpectedly, they behave differently on temporal oriented trees: we showed that Temporal Path Cover is polynomial-time solvable on temporal oriented trees (and a temporal version of Dilworth's theorem holds in this setting), while Temporally Disjoint Path Cover remains NP-hard for this class.

To prove our polynomial-time algorithm for Temporal Path Cover on temporal oriented trees, we have reduced the problem to Clique Cover in a static undirected graph, which turns out to be weakly chordal. This is a powerful technique, and the correspondence between the two problems is quite enlightening for the structure of temporal paths in an oriented tree. Nevertheless, it seems unlikely that this particular technique can be used on temporal digraph classes that are far from trees, as it was essential for the proof that any two vertices are joined by only one path in the underlying tree. However, this general technique could likely be applied in other temporal settings.

We do not know if our algorithm for treewidth and number of time-steps is optimal. In particular, can we obtain an FPT algorithm for Temporal Path Cover for this parameter? One could also explore other (structural) parameterizations of the problems.

We note that many of our results for Temporally Disjoint Path Cover also hold for its stricter vertex-disjoint version (note that a vertex-disjoint version of Temporally Disjoint Paths is studied in [24]), in particular, the NP-hardness result for restricted DAGs and the polynomial-time algorithms for rooted directed trees and oriented lines.

References

[1] Eleni C. Akrida, George B. Mertzios, Sotiris E. Nikoletseas, Christoforos L. Raptopoulos, Paul G. Spirakis, and Viktor Zamaraev. How fast can we reach a target vertex in stochastic temporal graphs? *J. Comput. Syst. Sci.*, 114:65–83, 2020.

[2] Eleni C. Akrida, George B. Mertzios, and Paul G. Spirakis. The temporal explorer who returns to the base. In Pinar Heggernes, editor, *Algorithms and Complexity - 11th International Conference, CIAC 2019, Rome, Italy, May 27-29, 2019, Proceedings*, volume 11485 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2019.

[3] Binh-Minh Bui-Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *Int. J. Found. Comput. Sci.*, 14(2):267–285, 2003.

[4] Manuel Cáceres. Minimum chain cover in almost linear time. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 31:1–31:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[5] Manuel Cáceres, Massimo Cairo, Brendan Mumey, Romeo Rizzi, and Alexandru I. Tomescu. Sparsifying, shrinking and splicing for minimum path cover in parameterized linear time. In *ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 359–376. SIAM, 2022.

[6] Manuel Cáceres, Brendan Mumey, Edin Husić, Romeo Rizzi, Massimo Cairo, Kristoffer Sahlin, and Alexandru I. Tomescu. Safety in multi-assembly via paths appearing in all path covers of a DAG. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(6):3673–3684, 2022.

[7] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *Int. J. Parallel Emergent Distributed Syst.*, 27(5):387–408, 2012.

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

[9] Yangjun Chen and Yibin Chen. On the graph decomposition. In *2014 IEEE Fourth International Conference on Big Data and Cloud Computing*, pages 777–784, 2014.

[10] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[11] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 4. Springer, 2015.

[12] Pradipta Kumar Das, Himansu Sekhar Behera, Prabir Kumar Jena, and Bijaya K. Panigrahi. An intelligent multi-robot path planning in a dynamic environment using improved gravitational search algorithm. *Int. J. Autom. Comput.*, 18(6):1032–1044, 2021.

[13] Robert P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51:161–166, 1950.

[14] Martin E. Dyer and Alan M. Frieze. Planar 3DM is NP-complete. *Journal of Algorithms*, 7(2):174–184, 1986.

[15] D. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[16] Delbert R Fulkerson. Note on Dilworth's decomposition theorem for partially ordered sets. In *Proceedings of the American Mathematical Society*, volume 7, pages 701–702, 1956.

[17] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[18] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57)*. North-Holland Publishing Co., NLD, 2004.

[19] Ryan B. Hayward. Weakly triangulated graphs. *J. Comb. Theory, Ser. B*, 39(3):200–208, 1985. doi:10.1016/0095-8956(85)90050-4.

[20] Ryan B. Hayward, Jeremy P. Spinrad, and R. Sritharan. Improved algorithms for weakly chordal graphs. *ACM Trans. Algorithms*, 3(2):14, 2007. doi:10.1145/1240233.1240237.

[21] Petter Holme. Modern temporal network theory: a colloquium. *European Physical Journal B*, 88(9), 2015.

[22] H. V. Jagadish. A compression technique to materialize transitive closure. *ACM Transactions on Database Systems*, 15(4):558–598, 1990.

[23] Naoyuki Kamiyama and Yasushi Kawase. On packing arborescences in temporal networks. *Inf. Process. Lett.*, 115(2):321–325, 2015.

[24] David Kempe, Jon M. Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. *J. Comput. Syst. Sci.*, 64(4):820–842, 2002.

[25] 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.

[26] 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.

[27] T. Kloks. *Treewidth, Computations and Approximations*. Springer, 1994.

[28] T. Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS 2021)*, pages 184–192, 2022.

[29] Pascal Kunz, Hendrik Molter, and Meirav Zehavi. In which graph structures can we efficiently find temporally disjoint paths and walks? In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 180–188. ijcai.org, 2023.

[30] Pascal Kunz, Hendrik Molter, and Meirav Zehavi. In which graph structures can we efficiently find temporally disjoint paths and walks? *arXiv preprint arXiv:2301.10503*, 2023.

[31] L. Lovász. Normal hypergraphs and the perfect graph conjecture. *Discrete Mathematics*, 2(3):253–267, 1972.

[32] Veli Mäkinen, Alexandru I. Tomescu, Anna Kuosmanen, Topi Paavilainen, Travis Gagie, and Rayan Chikhi. Sparse dynamic programming on dags with small width. *ACM Trans. Algorithms*, 15(2), feb 2019.

[33] Loris Marchal, Hanna Nagy, Bertrand Simon, and Frédéric Vivien. Parallel scheduling of DAGs under memory constraints. In *2018 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2018, Vancouver, BC, Canada, May 21-25, 2018*, pages 204–213. IEEE Computer Society, 2018.

[34] Andrea Marino and Ana Silva. Eulerian walks in temporal graphs. *Algorithmica*, 85(3):805–830, 2023.

[35] George B. Mertzios, Hendrik Molter, Malte Renken, Paul G. Spirakis, and Philipp Zschoche. The complexity of transitively orienting temporal graphs. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia*, volume 202 of *LIPIcs*, pages 75:1–75:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[36] Othon Michail. An introduction to temporal graphs: An algorithmic perspective. In Christos D. Zaroliagis, Grammati E. Pantziou, and Spyros C. Kontogiannis, editors, *Algorithms, Probability, Networks, and Games - Scientific Papers and Essays Dedicated to Paul G. Spirakis on the Occasion of His 60th Birthday*, volume 9295 of *Lecture Notes in Computer Science*, pages 308–343. Springer, 2015.

[37] Jérôme Monnot and Sophie Toulouse. The path partition problem and related problems in bipartite graphs. *Oper. Res. Lett.*, 35(5):677–684, 2007.

[38] Simeon C. Ntafos and S. Louis Hakimi. On path cover problems in digraphs and applications to program testing. *IEEE Transactions on Software Engineering*, SE-5(5):520–529, 1979.

[39] Jari Saramäki and Petter Holme, editors. *Temporal Network Theory*. Computational Social Sciences. Springer, Germany, October 2019.

[40] Jeremy P. Spinrad and R. Sritharan. Algorithms for weakly triangulated graphs. *Discret. Appl. Math.*, 59(2):181–191, 1995. doi:10.1016/0166-218X(93)E0161-Q.

[41] Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Roman Barták, and Eli Boyarski. Multi-agent pathfinding: Definitions, variants, and benchmarks. In Pavel Surynek and William Yeoh, editors, *Proceedings of the Twelfth International Symposium on Combinatorial Search, SOCS 2019, Napa, California, 16-17 July 2019*, pages 151–159. AAAI Press, 2019.

[42] Dawei Sun, Jingkai Chen, Sayan Mitra, and Chuchu Fan. Multi-agent motion planning from signal temporal logic specifications. *IEEE Robotics Autom. Lett.*, 7(2):3451–3458, 2022.

[43] 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.

[44] Haifeng Xu, Fei Fang, Albert Xin Jiang, Vincent Conitzer, Shaddin Dughmi, and Milind Tambe. Solving zero-sum security games in discretized spatio-temporal domains. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, page 1500–1506. AAAI Press, 2014.

[45] Yue Yin and Bo An. Efficient resource allocation for protecting coral reef ecosystems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 531–537. AAAI Press, 2016.

[46] Youzhi Zhang, Bo An, Long Tran-Thanh, Zhen Wang, Jiarui Gan, and Nicholas R. Jennings. Optimal escape interdiction on transportation networks. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 3936–3944. ijcai.org, 2017.