



Timeline Problems in Temporal Graphs: Vertex Cover vs. Dominating Set

Anton Herrmann  

Algorithmics and Computational Complexity, Technische Universität Berlin, Germany

Christian Komusiewicz  

Institute of Computer Science, Friedrich Schiller University Jena, Germany

Nils Morawietz  

Institute of Computer Science, Friedrich Schiller University Jena, Germany

LaBRI, Université de Bordeaux, Talence, France

Frank Sommer  

Institute of Logic and Computation, TU Wien, Austria

Institute of Computer Science, Friedrich Schiller University Jena, Germany

Abstract

A temporal graph is a finite sequence of graphs, called snapshots, over the same vertex set. Many temporal graph problems turn out to be much more difficult than their static counterparts. One such problem is TIMELINE VERTEX COVER (also known as $\text{MINTIMELINE}_\infty$), a temporal analogue to the classical VERTEX COVER problem. In this problem, one is given a temporal graph \mathcal{G} and two integers k and ℓ , and the goal is to cover each edge of each snapshot by selecting for each vertex at most k activity intervals of length at most ℓ each. Here, an edge uv in the i th snapshot is covered, if an activity interval of u or v is active at time i . In this work, we continue the algorithmic study of TIMELINE VERTEX COVER and introduce the TIMELINE DOMINATING SET problem where we want to dominate all vertices in each snapshot by the selected activity intervals.

We analyze both problems from a classical and parameterized point of view and also consider partial problem versions, where the goal is to cover (dominate) at least t edges (vertices) of the snapshots. With respect to the parameterized complexity, we consider the temporal graph parameters vertex-interval-membership-width (vimw) and interval-membership-width (imw). We show that all considered problems admit FPT-algorithms when parameterized by $\text{vimw} + k + \ell$. This provides a smaller parameter combination than the ones used for previously known FPT-algorithms for TIMELINE VERTEX COVER. Surprisingly, for $\text{imw} + k + \ell$, TIMELINE DOMINATING SET turns out to be easier than TIMELINE VERTEX COVER, by also admitting an FPT-algorithm, whereas the vertex cover version is NP-hard even if $\text{imw} + k + \ell$ is constant. We also consider parameterization by combinations of n , the vertex set size, with k or ℓ and parameterization by t . Here, we show for example that both partial problems are fixed-parameter tractable for t which significantly improves and generalizes a previous result for a special case of PARTIAL TIMELINE VERTEX COVER with $k = 1$.

2012 ACM Subject Classification Theory of computation \rightarrow Parameterized complexity and exact algorithms; Theory of computation \rightarrow Graph algorithms analysis

Keywords and phrases NP-hard problem, FPT-algorithm, interval-membership-width, Color coding

Digital Object Identifier 10.4230/LIPIcs.IPEC.2025.12

Related Version Some of the results of this work are also contained in the first author's Master's thesis [21].

Full Version: <https://doi.org/10.48550/arXiv.2510.08124>

Funding *Nils Morawietz:* Supported by the French ANR, project ANR-22-CE48-0001 (TEM-POGRAL).

Frank Sommer: Supported by the Alexander von Humboldt Foundation.



© Anton Herrmann, Christian Komusiewicz, Nils Morawietz, and Frank Sommer; licensed under Creative Commons License CC-BY 4.0

20th International Symposium on Parameterized and Exact Computation (IPEC 2025).

Editors: Akanksha Agrawal and Erik Jan van Leeuwen; Article No. 12; pp. 12:1–12:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

A crucial task in the management of wireless sensor networks is to monitor the network by selecting few dedicated sensors that can monitor themselves and other sensors which are close enough to have a direct wireless connection [27,30]. In graph-theoretic terms, this is the famous NP-hard DOMINATING SET problem where we say that a vertex dominates itself and all its neighbors and the task is to select few vertices of the graph such that every vertex is dominated by some selected vertex. In some applications of sensor networks, the network may change over time. Then, instead of a static graph G the input would be a *temporal graph* \mathcal{G} , consisting of a set of *snapshots* G_i , each reflecting the connections at timestep i . Moreover, in such a scenario each sensor could carry out the surveillance only for a bounded duration, for example due to limited battery capacities. Thus, instead of selecting few sensors to monitor the network, we would like to select, for each sensor, an active time interval of bounded length ℓ , during which it monitors itself and all its current neighbors. To make the model more general, we may also allow for each sensor to select up to k such active intervals, for example because each sensor carries k batteries. By calling a collection of active intervals for all vertices of the graph a *k -activity ℓ -timeline*, the scenario described above corresponds to the following problem.

TIMELINE DOMINATING SET (TIMELINE DS)

Input: A temporal graph \mathcal{G} and integers $k \geq 1, \ell \geq 0$.

Question: Is there a k -activity ℓ -timeline \mathcal{T} which dominates all temporal vertices of \mathcal{G} ?

For an example input and solution for TIMELINE DS, refer to Figure 1. As in the static DOMINATING SET problem, we may further generalize the problem to handle scenarios where we are not able to monitor the whole network over the full time period but instead we want to maximize the number of monitored sensors under the resource limitations.

TIMELINE PARTIAL DOMINATING SET (TIMELINE PDS)

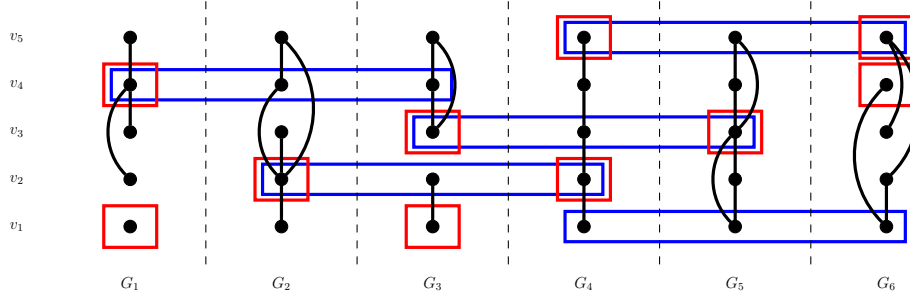
Input: A temporal graph \mathcal{G} and integers $k \geq 1, \ell \geq 0, t \geq 0$.

Question: Is there a k -activity ℓ -timeline \mathcal{T} which dominates at least t temporal vertices of \mathcal{G} ?

In this work, we initialize the study of TIMELINE (PARTIAL) DOMINATING SET in particular from a computational complexity perspective. As we show, TIMELINE (PARTIAL) DOMINATING SET is NP-hard. To cope with this intractability, we consider mostly the parameterized complexity of the problem, with a particular focus on structural parameterizations of temporal graphs.

The idea to consider timelines in a temporal graph setting is not new: Rozenshtein et al. [28] introduced it in TIMELINE VERTEX COVER,¹ the second main problem studied in this work.

¹ Rozenshtein et al. denote this problem as $\text{MINTIMELINE}_\infty$.



■ **Figure 1** The short red boxes represent a 2-activity 0-timeline dominating all temporal vertices and the long blue boxes represent a 1-activity 2-timeline covering all temporal edges. Observe that the interval length is defined in such a way that an interval starting and ending in the same snapshot has length 0.

TIMELINE VERTEX COVER (TIMELINE VC)

Input: A temporal graph \mathcal{G} and integers $k \geq 1, \ell \geq 0$.

Question: Is there a k -activity ℓ -timeline \mathcal{T} which covers all temporal edges of \mathcal{G} ?

For an example input and solution for TIMELINE VC, refer again to Figure 1. The model behind TIMELINE VC is that edges in a temporal graph arise only when at least one of their endpoints is active and the task of the problem is to provide an explanation of all edges such that the vertices are only active a few times, each time only for a short duration [28]. Dondi et al. [12] later analyzed the partial variant of the problem.

TIMELINE PARTIAL VERTEX COVER (TIMELINE PVC)

Input: A temporal graph \mathcal{G} and integers $k \geq 1, \ell \geq 0, t \geq 0$.

Question: Is there a k -activity ℓ -timeline \mathcal{T} which covers at least t temporal edges of \mathcal{G} ?

We continue the study of TIMELINE (PARTIAL) VERTEX COVER with two aims: First, some of the parameters considered in our study have not yet been studied for this problem. Second, we seek a comprehensive complexity overview for these two fundamental timeline problems, highlighting their similarities and differences.

Previous results. Rozenshtein et al. [28] showed that TIMELINE VC is NP-hard in general, but solvable in polynomial time for $k = 1$. Froese et al. [18] continued the algorithmic study of TIMELINE VC. They proved NP-hardness even if the input consists of at most three snapshots, $k = 2$ and $\ell = 0$ and studied the parameterized complexity with respect to different parameters. Froese et al. [18] showed fixed-parameter tractability for $n + k$ and W[1]-hardness for $n + \ell$. Herein, n is the number of vertices of the underlying graph which in turn is the static graph obtained by taking the union of all edge sets. Moreover, TIMELINE VC admits an XP-algorithm for n and is fixed-parameter tractable with respect to n if $\ell = 0$ [18]. Schubert [29] later analyzed the parameterized complexity of TIMELINE VC with respect to combinations of input parameters with structural parameters of the underlying graph. Finally, Dondi et al. [12] obtained two results for TIMELINE PVC with $k = 1$: First, it was

shown that this case is NP-hard, in contrast to TIMELINE VC. Second, Dondi et al. [12] gave FPT-algorithms for parameterization by the number of covered edges and by the number of uncovered edges.

Our results. We start off with a new hardness result for TIMELINE VC: We show that the problem remains NP-hard even if the total number of snapshots T , k , and ℓ are constant and additionally the maximum degree in every snapshot is at most one. Then, we show the NP-hardness of TIMELINE DS, again for the case that T , k , and ℓ are constant and the maximum snapshot degree is one.

We then study the parameterized complexity of the problems; an overview of the results is given in Table 1. As noted above, TIMELINE VC is fixed-parameter tractable with respect to $n + k$ [18]. Since n is a rather large parameter, Froese et al. [18] asked whether there are FPT-algorithms also for parameters that are smaller than n . We make progress in this direction by considering the two parameters *interval-membership-width* (imw) and *vertex-interval-membership-width* (vimw) which were introduced by Bumpus and Meeks [3]. The parameters imw and vimw are interesting in the sense that they are among the relatively few truly temporal structural graph parameters because they are sensitive to the order of the snapshots [3]. Informally, vimw can be defined as follows: We say the *lifetime* of a vertex is the time interval between its first and last non-isolated appearance in a snapshot. For a timestep i , the bag of i is the set of vertices whose lifetime contains timestep i , and vimw is the size of the largest bag of \mathcal{G} .

We first show that TIMELINE PVC is fixed-parameter tractable with respect to $\text{vimw} + k + \ell$. To further improve on this, we introduce a hierarchy of parameters with non-increasing size. The idea, inspired by the h -index of static graphs [16], is that a temporal graph may have only few large bags. In that case, we would rather parameterize by the size of the x th largest bag, denoted by $\text{vimw}[x]$, for some $x > 1$. With this definition, we have $\text{vimw} = \text{vimw}[1]$ and $\text{vimw}[x] \geq \text{vimw}[x + 1]$ for all $x \in [T]$. We show that TIMELINE PVC is fixed-parameter tractable for $\text{vimw}[x] + k + \ell$ with $x = k \cdot (\ell + 1)$. Informally, this means that the larger k and ℓ get, the more large bags can be ignored for the parameter. We then consider TIMELINE (PARTIAL) DOMINATING SET in the same setting. We first showing fixed-parameter tractability of TIMELINE PDS for $\text{vimw} + k + \ell$. Then we show that for TIMELINE DS it can be improved to fixed-parameter tractability for $\text{vimw}[x] + k + \ell$ for $x = \ell + 2$ while for TIMELINE PDS parameterization by $\text{vimw}[2] + k + \ell$ is not useful. The latter result is obtained by showing NP-hardness of the extremely restricted special case when there are two snapshots, one of which is edgeless, $k = 1$ and $\ell = 0$.

Afterwards, we consider the interval-membership-width (imw). Informally, this is the edge-version of vimw, that is, the bag at timestep i contains the edges which occur at timestep i and those that occur both before and after i . Note that imw can be considered to be a smaller parameter than vimw: For all instances, we have $\text{vimw} \in \Omega(\sqrt{\text{imw}})$ and there are instances with constant values of imw and unbounded values of vimw. We show that TIMELINE DS is fixed-parameter tractable for the parameter $\text{imw} + k + \ell$ while all three other problems are NP-hard for constant values of $\text{imw} + k + \ell$. Given the latter hardness results, we refrain from analyzing a hierarchy of imw-based parameters. Altogether, the results show that in our setting, vimw is a much more powerful parameter than imw.

We then conclude by considering the more standard parameters n , k , ℓ , and t , where t denotes the number of vertices to dominate or edges to cover, respectively. For TIMELINE DS, we show fixed-parameter tractability for $n + k$ and W[1]-hardness for $n + \ell$, thus showing that the complexity is similar to the one of TIMELINE VC. Finally, we show that TIMELINE PDS

■ **Table 1** Overview of our results on the classic and parameterized complexity for TIMELINE (PARTIAL) VERTEX COVER and TIMELINE (PARTIAL) DOMINATING SET. Herein, vimw denotes the vertex-interval-membership-width, imw denotes the interval-membership-width, and vimw[x] denotes the size of the x -largest bag in the vertex-interval-membership sequence. Recall that vimw = vimw[1].

Parameter	TIMELINE VC	TIMELINE PVC	TIMELINE DS	TIMELINE PDS
$\text{vimw} + k + \ell$	FPT Thm. 4.1	FPT Thm. 4.1	FPT Thm. 4.3	FPT Thm. 4.3
$\text{imw} + k + \ell$	NP-h Thm. 4.7	NP-h Thm. 4.7	FPT Thm. 4.6	NP-h Thm. 4.8
$k + \ell + \text{vimw}[x]$	$x = k(\ell + 1) + 1$ FPT Prop. 4.2	$x = k(\ell + 1) + 1$ FPT Prop. 4.2	$x = \ell + 2$ FPT Prop. 4.4	$x = 2$ NP-h Thm. 4.5
$n + k$	FPT [18, Thm. 8]	?	FPT Thm. 5.1	?
$n + \ell$	W[1]-h [18, Thm. 12]	W[1]-h [18, Thm. 12]	W[1]-h Thm. 5.2	W[1]-h Thm. 5.2
t	—	FPT Thm. 5.4	—	FPT Thm. 5.4

and TIMELINE PVC can be solved in $2^{\mathcal{O}(t)} \cdot (n + T)^{\mathcal{O}(1)}$ time. This improves and extends a previous result of Dondi et al. [12, Thm. 6] who provided an algorithm for TIMELINE PVC with $k = 1$ that has running time $2^{t \cdot \log(t)} \cdot (n + T)^{\mathcal{O}(1)} = t^t \cdot (n + T)^{\mathcal{O}(1)}$.

Proofs of statements marked with (\star) are deferred to the related full version.

Further related work. Akrida et al. [1] introduced the problems TEMPORAL VERTEX COVER (TVC) and SLIDING WINDOW TEMPORAL VERTEX COVER (SW-TVC), where the goal is to find a minimum number of temporal vertices that cover the edges of the temporal graph. In TVC each edge needs to be covered in at least one of the snapshots and in SW-TVC, the input contains an integer Δ , and the aim is to cover every edge at least once at every Δ consecutive time steps. Akrida et al. [1] showed that TVC remains NP-hard when the underlying graph is a star graph. Moreover, they studied the approximability of both problems, provided (S)ETH-based running time lower bounds and designed an exact dynamic programming algorithm with exponential running time. The research on these problems was then continued by Hamm et al. [20], who showed that TVC is solvable in polynomial time if the underlying graph is a path or cycle while SW-TVC remains NP-hard in this case. Besides that, they provided several (exact and approximation) algorithms for the sliding window variant. The TEMPORAL DOMINATING SET (TDS) problem is defined analogously, here the task is to dominate every vertex in at least one of the snapshots of the temporal graph by selecting few temporal vertices. TDS is NP-hard in very restricted cases, for example when the maximum degree in every snapshot is 2 [22]. There are further temporal variants of VERTEX COVER and DOMINATING SET which have been discussed and studied. These include a multistage variant of VERTEX COVER [17], a reachability-based variant for DOMINATING SET [23] and different conditions on how and when a vertex should be dominated [4, 31].

2 Preliminaries

We denote the set of integers $\{i, i+1, \dots, j-1, j\}$ by $[i, j]$, and $[1, i]$ simply by $[i]$. Given a set V and an integer $k > 0$, we write $\binom{V}{k}$ for the collection of all size- k subsets of V .

Graph theory. A (static) graph $G = (V, E)$ consists of a set of *vertices* V and a set of edges $E \subseteq \binom{V}{2}$. We denote by $V(G)$ and $E(G)$ the vertex and edge set of G , respectively. Furthermore, we let $n := |V(G)|$ and $m := |E(G)|$. For an edge $\{u, v\}$ we write uv and call u and v *endpoints* of the edge. Further, we say that the vertex $v \in V$ is *incident* with $e \in E$ if $v \in e$. If $uv \in E$, then u and v are *adjacent* and they are *neighbors* of each other. The set $N_G(v) := \{u \in V : u \text{ and } v \text{ are adjacent}\}$ is the (*open*) *neighborhood* of v . Moreover, $N_G[v] := N_G(v) \cup \{v\}$ is the *closed neighborhood* of v . For a set of vertices S , we let $N_G[S] := \bigcup_{s \in S} N_G[s]$ and $N_G(S) = N_G[S] \setminus S$. By $\deg_G(v) := |N_G(v)|$ we denote the *degree* of v . If $\deg_G(v) = 0$, we say that v is *isolated*. The *maximum degree* is $\Delta(G) := \max_{v \in V} \deg_G(v)$. For vertex sets S and T , we let $E_G(S, T) := \{uv \in E(G) : u \in S \text{ and } v \in T\}$ and we let $E(S) := E(S, S)$. A graph G is *bipartite* if $V(G)$ can be partitioned into two sets S and T such that $E(G) = E(S, T)$. For more details on graph theory, we refer to the book by Diestel [8].

Temporal graphs. A *temporal graph* \mathcal{G} is a finite sequence of graphs (G_1, \dots, G_T) which all have the same set of vertices $V(\mathcal{G}) := V(G_1) = \dots = V(G_T)$. The graphs G_i are called *snapshots*, T is the *lifetime* of \mathcal{G} , and $i \in [T]$ is a *time step*. We may write V instead of $V(\mathcal{G})$ and E_i instead of $E(G_i)$. Moreover, for $u, v \in V$ and $i \in [T]$, (v, i) is a *temporal vertex* and $(\{u, v\}, i) = (uv, i)$ with $uv \in E_i$ is a *temporal edge* in snapshot G_i . We say edge e *appears* in G_i if $e \in E_i$. Moreover, we say that a vertex v is *incident* with the temporal edge (e, i) if $v \in e$. For a temporal vertex (v, i) , we define the (*open*) *neighborhood* by $N_{\mathcal{G}}((v, i)) := N_{G_i}(v)$ and the *closed neighborhood* by $N_{\mathcal{G}}[(v, i)] := N_{G_i}[v]$. By $G_{\downarrow}(\mathcal{G}) := (V_{\downarrow} := V(\mathcal{G}), E_{\downarrow} := \bigcup_{i=1}^T E_i)$ we denote the *underlying graph* of \mathcal{G} . We let $\Delta_i(\mathcal{G}) = \Delta(G_i)$. Furthermore, by $\Delta_{\max}(\mathcal{G}) := \max_{i=1}^T \Delta_i(\mathcal{G})$ we denote the *maximum snapshot degree* of \mathcal{G} . If $E_i = \emptyset$ we say that G_i is *empty*. We may drop the subscript $\cdot_{\mathcal{G}}$ when it is clear from context.

Parameter definitions. We give the precise definitions of the parameters related to the (vertex)-interval-membership width: vimw , $\text{vimw}[x]$, and imw .

Let $\mathcal{G} = (G_1, \dots, G_T)$ be a temporal graph. The *vertex-interval-membership sequence* of \mathcal{G} is the sequence $(F_i)_{i \in [T]}$ of vertex-subsets $F_i \subseteq V(\mathcal{G})$ (called *bags*), where each F_i is defined as

$$F_i := \{v \in V(\mathcal{G}) : \exists p, q \in [T] \text{ such that } p \leq i \leq q, \deg_{G_p}(v) \geq 1 \text{ and } \deg_{G_q}(v) \geq 1\}.$$

The *vertex-interval-membership-width* of \mathcal{G} , denoted by $\text{vimw}(\mathcal{G})$, is the maximum size of a bag in the vertex-interval-membership sequence, that is, $\text{vimw}(\mathcal{G}) := \max\{|F_i| : i \in [T]\}$.

For given $x \in [T]$ we introduce the parameter $\text{vimw}[x]$ as the size of the x th largest bag of the vertex-interval-membership sequence. Formally, let $(F_{i_1}, \dots, F_{i_T})$ be an ordering of $(F_i)_{i \in [T]}$ such that $|F_{i_1}| \geq \dots \geq |F_{i_T}|$. Then we define

$$\text{vimw}[x](\mathcal{G}) := |F_{i_x}|.$$

Note that $\text{vimw}[i]$ can be much smaller than $\text{vimw}[1] = \text{vimw}$: for example consider a temporal graph where every vertex only appears non-isolated for a short period of time. If all snapshots have only a few edges, then vimw is small. However, a single snapshot with many edges is enough to make vimw arbitrarily large. If only a few such snapshots exist, then $\text{vimw}[i]$ is already much smaller than vimw for small i .

The *interval-membership sequence* of \mathcal{G} is the sequence $(F_i)_{i \in [T]}$ of edge-subsets $F_i \subseteq E_\downarrow$ (again called *bags*), where each F_i is defined as

$$F_i := \{e \in E_\downarrow : \exists p, q \in [T] \text{ such that } p \leq i \leq q \text{ and } e \in E_p \cap E_q\}.$$

The *interval-membership-width* of \mathcal{G} , denoted by $\text{imw}(\mathcal{G})$, is the maximum size of a bag in the interval-membership sequence, that is, $\text{imw}(\mathcal{G}) := \max\{|F_i| : i \in [T]\}$.

We remark that $\text{vimw} = \Omega(\sqrt{\text{imw}})$ and that there exist temporal graphs with $\text{imw} = 1$ but arbitrary large vimw [3]. Finally, note that the vertex-interval-membership sequence and the interval-membership sequence can be computed in polynomial time with respect to the input size.

Timelines. Given a temporal graph $\mathcal{G} = (G_1, \dots, G_T)$, an *activity interval* of a vertex v is a triple $(v, a, b) \in V(\mathcal{G}) \times [T] \times [T]$ such that $a \leq b$. We say a that is the *starting time* and b is the *end time* of an activity interval (v, a, b) . The *length* of the activity interval is defined by $b - a$. A *k-activity timeline* of the temporal graph \mathcal{G} is a set of activity intervals $\mathcal{T} \subseteq \{(v, a, b) \in V(\mathcal{G}) \times [T] \times [T] : a \leq b\}$, which contains at most k activity intervals for each vertex. A *k-activity ℓ -timeline* is a k -activity timeline in which each activity interval has length at most ℓ . Given a k -activity timeline \mathcal{T} , a vertex v is called *active* in snapshot G_i , if there exists $(v, a, b) \in \mathcal{T}$ such that $a \leq i \leq b$. A k -activity timeline \mathcal{T} *dominates* the temporal vertex (v, i) if $v \in N_{G_i}[\{u : u \text{ is active in } G_i\}]$. A k -activity timeline \mathcal{T} *covers* the temporal edge (e, i) if at least one of the two endpoints of e is active in snapshot G_i .

Parameterized complexity. Let $L \subseteq \Sigma^*$ be a computational problem specified over some alphabet Σ and let $p : \Sigma^* \rightarrow \mathbb{N}$ be a *parameter*, that is, p assigns to each instance of L an integer parameter value (which we simply denote by p if the instance is clear from the context). We say that L is *fixed-parameter tractable (FPT) with respect to p* if L can be decided in $f(p) \cdot |I|^{\mathcal{O}(1)}$ time where $|I|$ is the length of the input. The corresponding hardness concept related to fixed-parameter tractability is $W[t]$ -hardness, $t \geq 1$; if a problem L is $W[t]$ -hard with respect to p , then L is assumed to *not* be fixed-parameter tractable. Moreover, L is *slice-wise polynomial (XP) with respect to p* if L can be decided in $f(p) \cdot |I|^{g(p)}$ time. Let (I, p) and (I', p') be two instances of L . A *reduction to a (problem) kernel for L* is a polynomial-time algorithm that computes an instance (I', p') such that

- $p' + |I'| \leq g(p)$, and
- (I, p) is a yes-instance of L if and only if (I', p') is a yes-instance of L .

The instance (I', p') is referred to as *the problem kernel* and g is the *size* of the kernel. Furthermore, if g is a polynomial, we say that the kernel is a *polynomial problem kernel*. For more details about parameterized complexity we refer to the standard monographs [7, 14].

3 NP-Hardness Results

We show NP-hardness for TIMELINE VC and TIMELINE DS even when they are restricted to temporal graphs with a maximum snapshot degree of 1 and constant lifetime T , interval number k , and interval length ℓ . First, we present the reduction for TIMELINE VC.

► **Theorem 3.1.** *TIMELINE VC is NP-hard even if $T = 23, k = 2, \ell = 4$, and the maximum snapshot degree is one.*

Proof. We reduce from the NP-hard problem 3-COLORING on graphs with maximum degree four [19]. In 3-COLORING the input is a graph and the question is whether the vertices can be colored with three colors such that no two adjacent vertices have the same color.

Intuition. The basic idea is to construct a temporal graph with three parts C_1, C_2 and C_3 (call them *color blocks*), each consisting of five snapshots and representing one of the three colors. The part in which a vertex v is *not active* corresponds to the assigned color of v . As each of the three parts will contain every edge of G and only vertices of degree at most one, this ensures that all neighbors are active in the part where v is not active. Hence, they are assigned a different color. The basic idea for encoding a given graph G into snapshots of maximum degree one is to find a proper edge coloring with five colors and split the edge set with respect to this coloring, which can be done in polynomial time via Vizing's theorem. Our aim is that no activity interval of any vertex can hit more than one color block. To ensure this, we add sufficiently many empty snapshots between any two parts.

Construction. Let $(G = (V = \{v_1, \dots, v_n\}, E))$ be an instance of 3-COLORING on graphs with maximum degree four. We construct the following temporal graph \mathcal{G} where $V(\mathcal{G}) = V$.

The temporal graph \mathcal{G} consists of 23 snapshots. It can be seen as a sequence of five parts, namely C_1, H_1, C_2, H_2, C_3 . Parts H_1 and H_2 consist of 4 empty snapshots and each of C_1, C_2, C_3 consists of 5 snapshots. Thus, snapshots 6 to 9 and 15 to 18 are empty. We denote H_1 and H_2 as the *empty blocks* and we call C_1, C_2 and C_3 *color blocks* as they correspond to the three colors. In the following we formally define the snapshots of the color blocks. All three color blocks C_1, C_2 and C_3 will be identical. Hence, we only formally define C_1 , that is, snapshots 1 to 5.

Here we exploit the fact that the graph G has maximum degree four, because this implies that there exists a proper edge coloring with at most five colors due to Vizing's theorem [32], which can be computed in polynomial time [25]. Let $E(G) = F_1 \cup \dots \cup F_5$ be the partition of $E(G)$ into five colors of some proper edge coloring. Now, snapshot j contains exactly the edges of F_j . Finally, observe that by construction, the maximum degree in each snapshot is at most one. By setting $k = 2$ and $\ell = 4$, we obtain the TIMELINE VC instance (\mathcal{G}, k, ℓ) .

Correctness. We show that G is 3-colorable if and only if \mathcal{G} has a 2-activity 4-timeline \mathcal{T} covering all temporal edges of \mathcal{G} . In the following we say a vertex v is *active* during C_j for $j \in [3]$ to indicate that an activity interval of v starts at the first time step and ends at the last time step of the corresponding part.

(\Rightarrow) Let $\phi : V(G) \rightarrow [3]$ be a proper 3-coloring of G . We construct an activity timeline of \mathcal{G} which covers all temporal edges. For each $i \in [n]$, the vertex $v \in V(G)$ is active in the two color blocks, which do not correspond to the assigned color of v . Clearly, this yields a 2-activity timeline \mathcal{T} . It remains to argue that each temporal edge is covered by \mathcal{T} . Since ϕ is a proper 3-coloring, both endpoints of each edge $uv \in E(G)$ have a different color. Moreover, since in color block C_j all vertices are active which do not have color j , we conclude that all temporal edges of \mathcal{G} are covered by \mathcal{T} .

(\Leftarrow) Let \mathcal{T} be a 2-activity 4-timeline covering all temporal edges of \mathcal{G} . Observe that since parts H_1 and H_2 corresponding to snapshots 6 to 9 and 15 to 18 are empty, no activity interval of any vertex can hit more than one color block. Consequently, since $k = 2$, for each vertex v there exists at least one color $j \in [3]$ such that v is not active during any snapshot of color block C_j . Let $\phi(v) \in [3]$ be such a color. Note that if $\phi(v)$ is not unique, we pick any color fulfilling the property. We now argue that ϕ is a proper 3-coloring of G .

Assume towards a contradiction that this is not the case, that is, there exists at least one edge $uw \in E(G)$ such that both endpoints u and w have the same color j . This implies that both u and w are not active during C_j . But since exactly one of the 5 snapshots corresponding to color block C_j contains the edge uw , timeline \mathcal{T} is not covering all temporal edges of \mathcal{G} , a contradiction. Thus, ϕ is a proper 3-coloring of G . \blacktriangleleft

Next, we extend the ideas of this reduction to TIMELINE DS for which the reduction is more involved since we also need to deal with isolated temporal vertices.

► **Theorem 3.2** (\star). *TIMELINE DS is NP-hard even if $T = 35, k = 3, \ell = 6$ and the maximum snapshot degree is one.*

4 The Influence of Membership-Width Based Parameters

Now, we investigate the membership-width based parameters vimw and imw . More precisely, we study the parameter combinations $\text{vimw} + k + \ell$ and $\text{imw} + k + \ell$. These combinations are motivated by the fact that TIMELINE VC and TIMELINE DS are $W[1]$ -hard with respect to $n + \ell$; for TIMELINE VC this follows from previous work [18] and for TIMELINE DS, we will show this in Section 5.

4.1 The Parameter Vertex-Interval-Membership-Width

In this section, we provide FPT-algorithms with respect to $\text{vimw} + k + \ell$ for TIMELINE PVC and TIMELINE PDS. Moreover, we show that vimw can be replaced by smaller parameters $\text{vimw}[x]$ for all problems except TIMELINE PDS (the precise value of x differs for TIMELINE PVC and TIMELINE DS). Simultaneously, these algorithms also are XP-algorithms for TIMELINE PVC and TIMELINE PDS parameterized by vimw alone. Initially, we focus on TIMELINE (PARTIAL) VERTEX COVER and we show that TIMELINE PVC admits an FPT-algorithm for $\text{vimw} + k + \ell$.

► **Theorem 4.1** (\star). *TIMELINE PVC is solvable in $((k+1)(\ell+2))^{2 \cdot \text{vimw}} \cdot (n+T)^{\mathcal{O}(1)}$ time.*

Proof. Let $(\mathcal{G} = (G_1, \dots, G_T), k, \ell, t)$ be an instance of TIMELINE PVC. Further, suppose that $T \geq k \cdot (\ell + 1) + 1$ (otherwise, it is a trivial instance) and that the underlying graph G_\downarrow contains no isolated vertex v . Otherwise, since v does not cover any temporal edge, v can be safely removed and consequently t remains unchanged. Moreover, we assume that each vertex v is contained in at least $k \cdot (\ell + 1) + 1$ bags (recall that these are consecutive) because otherwise we can simply greedily pick k activity intervals of v which contain all snapshots where v is non-isolated and thus we cover all temporal edges incident with v . Thus, it is safe to remove v from \mathcal{G} and reduce t by the number of temporal edges incident with v over all snapshots.

The idea is to use dynamic programming over the lifetime of the temporal graph. Each table entry corresponds to the maximal number of covered temporal edges where a specific set of vertices in the bag is active. In other words, the dynamic program keeps track of the activity of the vertices which are contained in the bag F_i of the currently considered time step $i \in [T]$. While iterating over the lifetime of the temporal graph, we need to ensure that the activities at a time step are compatible with the activities at the neighboring time steps. Moreover, for a vertex v let i and j be the index of the smallest and largest snapshot where v is incident to an edge, respectively. By definition, $v \in F_x$ for any $i \leq x \leq j$. Furthermore, since v is isolated in all snapshots G_x where $x < i$ or $x > j$, it is safe to assume that all k activity intervals of v are used when v is contained in the bags F_x .

12:10 Timeline Problems in Temporal Graphs: Vertex Cover vs. Dominating Set

Recall that the vertex-interval-membership sequence can be computed in polynomial time. Since we assumed that the underlying graph contains no isolated vertex, it follows that each vertex is contained in at least one bag of the sequence.

Table Definition. We denote $\overleftarrow{F}_i := F_1 \cup \dots \cup F_i$ and $\overline{F}_i := \overleftarrow{F}_i \setminus F_i$. In other words, \overline{F}_i is the set of vertices which are isolated in all snapshots G_i, \dots, G_T since we assumed that G_\downarrow contains no isolated vertices. Moreover, for $i \in [T]$ we define functions $\text{curr} : F_i \rightarrow [0, k]$, and $\text{pos} : F_i \rightarrow [0, \ell + 1]$. The functions curr and pos indicate for each vertex $v \in F_i$ whether an activity interval of v is active during snapshot G_i . More precisely, the function value $\text{curr}(v)$ determines the number of the current activity interval of v and the function value $\text{pos}(v)$ determines the position in this activity interval. Here, $\text{curr}(v) = 0$ means that the current time step is before the first activity interval and $\text{pos}(v) = 0$ means that v is currently not active, that is, the $\text{curr}(v)$ -th activity interval of v is already over and the next one has not started yet. Now the entry $\text{DP}[i, \text{curr}, \text{pos}]$ contains the maximum number of covered temporal edges in the first i snapshots G_1, \dots, G_i , while i is at the $\text{pos}(v)$ -th position of the $\text{curr}(v)$ -th activity interval of $v \in F_i$.

The formal definition of the table now is

$$\text{DP}[i, \text{curr}, \text{pos}] \triangleq \max_{\mathcal{T}} |\{(e, j) \in E_\downarrow \times [i] : (e, j) \text{ is covered by } \mathcal{T}\}|,$$

where the maximum is taken over all k -activity ℓ -timelines

$$\mathcal{T} \subseteq (\overline{F}_i \times [i - 1] \times [T]) \cup (F_i \times [i] \times [T]),$$

such that for each $v \in F_i$

- (i) there are at most $\text{curr}(v)$ activity intervals of v in \mathcal{T} ,
- (ii) $\text{pos}(v) \leq i$, and
- (iii) $(v, i - \text{pos}(v) + 1, b) \in \mathcal{T}$ for $b = \min(i - \text{pos}(v) + 1 + \ell, T)$ if $\text{pos}(v) > 0$.

We need Conditions (i)-(iii) for the following reasons: Condition (i) ensures that each vertex has at most k activity intervals. Moreover, condition (ii) ensures that no activity interval starts before time step one. Finally, condition (iii) ensures that each activity interval has length ℓ , except if the remaining number of snapshots is too small.

Initialization. For all $\text{curr} : F_1 \rightarrow [0, k]$ and $\text{pos} : F_1 \rightarrow [0, 1]$, for which $\text{curr}(v) = 0$ implies $\text{pos}(v) = 0$, the table is initialized by

$$\text{DP}[1, \text{curr}, \text{pos}] = |E_{G_1}(\{v \in F_1 : \text{pos}(v) = 1\}, F_1)|.$$

Note that $\overline{F}_1 = \emptyset$ and therefore only activity timelines $\mathcal{T} \subseteq F_1 \times [1] \times [T]$ are considered in the definition of $\text{DP}[1, \text{curr}, \text{pos}]$. The initialization is correct, since the table entry $\text{DP}[1, \text{curr}, \text{pos}]$ contains the number of temporal edges which are covered by the active vertices from F_1 in G_1 . Observe that the other endpoints of the covered edges are also contained in F_1 by definition.

Recurrence. The remaining table entries are computed recursively. Intuitively, $\text{DP}[i, \text{curr}, \text{pos}]$ is computed by trying all activity interval choices for the vertices of F_{i-1} , such that there is no conflict with curr and pos . Formally, we set

$$\text{DP}[i, \text{curr}, \text{pos}] = \max_{\text{curr}', \text{pos}'} \text{DP}[i - 1, \text{curr}', \text{pos}'] + |E_{G_i}(\{v \in F_i : \text{pos}(v) \geq 1\}, F_i)|$$

where the maximum is taken over all $\text{curr}' : F_{i-1} \rightarrow [0, k]$ and $\text{pos}' : F_{i-1} \rightarrow [0, \ell + 1]$ which are *compatible* with curr and pos . Informally, curr' and pos' are *compatible* with curr and pos if they provide a correct extension of the activities of $F_{i-1} \cap F_i$ at time step i to time

step $i - 1$. Formally, four conditions need to be fulfilled: First, if an interval of v is already active at time step i , that is, if $\text{pos}(v) \geq 2$, then an activity interval of v is already active during time step $i - 1$. This can be formalized as follows.

$$\text{If } \text{pos}(v) \geq 2, \text{ then } \text{curr}'(v) \leq \text{curr}(v) \text{ and } \text{pos}'(v) = \text{pos}(v) - 1. \quad (1)$$

Second, if a new activity interval of v starts at time step i either v was inactive at time step $i - 1$ or the previous activity interval of v ended at time step $i - 1$. This can be formalized as follows.

$$\text{If } \text{curr}(v) > 1 \text{ and } \text{pos}(v) = 1, \text{ then } \text{curr}'(v) \leq \text{curr}(v) - 1 \text{ and } \text{pos}'(v) \in \{\min(i - 1, \ell + 1), 0\}. \quad (2)$$

Third, if the first activity interval of v starts at time step i , then in the previous time step $i - 1$ vertex v cannot be active. This can be formalized as follows.

$$\text{If } \text{curr}(v) = 1 \text{ and } \text{pos}(v) = 1, \text{ then } \text{curr}'(v) = 0 \text{ and } \text{pos}'(v) = 0. \quad (3)$$

Finally, if v is not active during time step i , either v is also not active during time step $i - 1$ or an activity interval of v ended at time step $i - 1$. This can be formalized as follows.

$$\text{If } \text{curr}(v) \geq 1 \text{ and } \text{pos}(v) = 0, \text{ then } \text{curr}'(v) \leq \text{curr}(v) \text{ and } \text{pos}'(v) \in \{\min(i - 1, \ell + 1), 0\}. \quad (4)$$

The minimum in Conditions 2 and 4 ensures that the activity intervals do not have a starting point smaller than one.

Correctness. We show the correctness of the computation by proving inequalities in both directions.

(\leq) Suppose the maximum in the definition of the table entry $\text{DP}[i, \text{curr}, \text{pos}]$ is attained for \mathcal{T} and assume that no activity intervals from \mathcal{T} overlap. We define \mathcal{T}' by taking all activity intervals from \mathcal{T} which are relevant for a table entry of $i - 1$. Formally, we define

$$\begin{aligned} \mathcal{T}' := & \{(v, a, b) \in \mathcal{T} : a \leq i - 2, v \in \overline{F}_{i-1}\} \\ & \cup \{(v, a, b) \in \mathcal{T} : a \leq i - 1, v \in F_{i-1}\}, \end{aligned}$$

and, in accordance with \mathcal{T}' , functions $\text{curr}' : F_{i-1} \rightarrow [0, k], \text{pos}' : F_{i-1} \rightarrow [0, \ell + 1]$ by

$$\begin{aligned} \text{curr}'(v) &:= |\{(v, a, b) \in \mathcal{T}' : a, b \in [T]\}| \\ \text{pos}'(v) &:= \begin{cases} i - a & \text{if } (v, a, b) \in \mathcal{T}' \text{ with } a \leq i \leq b \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Note that $\mathcal{T}' \subseteq \mathcal{T}$. We show that curr' and pos' satisfy Conditions (1)–(4). This means that the table entry $\text{DP}[i - 1, \text{curr}', \text{pos}']$ is considered in the maximum on the right side of the recursive computation. By definition of curr' and pos' it is clear that \mathcal{T}' is then considered in the definition of the table entry $\text{DP}[i - 1, \text{curr}', \text{pos}']$. Now let $v \in F_{i-1} \cap F_i$.

- (1) If $\text{pos}(v) \geq 2$, then $(v, a, b) \in \mathcal{T}$ for some $a \leq i - 1, b \in [T]$ and by definition of \mathcal{T}' , each such activity interval of v in \mathcal{T} is also contained in \mathcal{T}' . Consequently, we have $\text{curr}'(v) \leq \text{curr}(v)$ and $\text{pos}'(v) = \text{pos}(v) - 1$.
- (2) If $\text{curr}(v) > 1$ and $\text{pos}(v) = 1$, then $(v, i, b) \in \mathcal{T} \setminus \mathcal{T}'$ for some $b \in [T]$. So the activity timeline \mathcal{T}' contains at most $\text{curr}(v) - 1$ activity intervals of v and therefore $\text{curr}'(v) \leq \text{curr}(v) - 1$. Now either $(v, a, i - 1) \in \mathcal{T}'$ for some $a \in [i - 1]$ or v is not active in G_{i-1} with respect to the activity intervals from \mathcal{T}' . This means that either $\text{pos}'(v) = \min(i - 1, \ell + 1)$ or $\text{pos}'(v) = 0$.

12:12 Timeline Problems in Temporal Graphs: Vertex Cover vs. Dominating Set

- (3) If $\text{curr}(v) = 1$ and $\text{pos}(v) = 1$, then $(v, i, b) \in \mathcal{T} \setminus \mathcal{T}'$ for some $b \in [T]$. Consequently, \mathcal{T}' contains no activity interval of v and therefore $\text{curr}'(v) = \text{pos}'(v) = 0$.
- (4) If $\text{curr}(v) \geq 1$ and $\text{pos}(v) = 0$, then either $(v, a, i-1) \in \mathcal{T}'$ for some $a \in [i-1]$ or v is not active in G_{i-1} with respect to the activity intervals from \mathcal{T}' . So by definition $\text{curr}'(v) \leq \text{curr}(v)$ and either $\text{pos}'(v) = \min(i-1, \ell+1)$ or $\text{pos}'(v) = 0$.

Now consider the set \mathcal{A} of temporal edges until time step i that are covered by \mathcal{T} and the set \mathcal{A}' of temporal edges until time step $i-1$ that are covered by \mathcal{T}' . There are two cases for a temporal edge $(vw, j) \in \mathcal{A} \setminus \mathcal{A}'$.

Case 1: $j = i$. Since the temporal edge (vw, j) is covered by \mathcal{T} we can conclude that $vw \in E_{G_i}(\{v \in F_i : \text{pos}(v) \geq 1\}, F_i)$.

Case 2: $j < i$. We have $(vw, j) \in \binom{F_{i-1}}{2} \times [i-1]$. This implies the existence of an activity interval (v, a, b) or (w, a, b) in \mathcal{T} with $a \leq i-1$. However, then the activity interval is by definition contained in \mathcal{T}' and consequently $(vw, j) \in \mathcal{A}'$.

Hence, any temporal edge in $\mathcal{A} \setminus \mathcal{A}'$ is contained in $E_{G_i}(\{v \in F_i : \text{pos}(v) \geq 1\}, F_i)$. We thus have the desired inequality

$$\begin{aligned} \text{DP}[i, \text{curr}, \text{pos}] &= |\mathcal{A}| \leq |\mathcal{A}'| + |E_{G_i}(\{v \in F_i : \text{pos}(v) \geq 1\}, F_i)| \\ &\leq \text{DP}[i-1, \text{curr}', \text{pos}'] + |E_{G_i}(\{v \in F_i : \text{pos}(v) \geq 1\}, F_i)|. \end{aligned}$$

(\geq) Suppose the maximum on the right side of the computation is attained for $\text{curr}', \text{pos}'$ and the maximum in the definition of $\text{DP}[i-1, \text{curr}', \text{pos}']$ is attained for \mathcal{T}' . We define \mathcal{T} by extending \mathcal{T}' in the following way: For $v \in F_{i-1} \cap F_i$ the activity interval $(v, i, \min(i+\ell, T))$ is added if and only if $\text{pos}(v) = 1$. These new intervals together with the already active ones ($\text{pos}(v) \geq 2$) cover all edges of $E_{G_i}(\{v \in F_i : \text{pos}(v) \geq 1\}, F_i)$. Hence, the total number of edges covered by \mathcal{T} is $\text{DP}[i-1, \text{curr}', \text{pos}'] + |E_{G_i}(\{v \in F_i : \text{pos}(v) \geq 1\}, F_i)|$.

Because \mathcal{T} is considered in the definition of $\text{DP}[i, \text{curr}, \text{pos}]$, we have

$$\text{DP}[i, \text{curr}, \text{pos}] \geq \text{DP}[i-1, \text{curr}', \text{pos}'] + |E_{G_i}(\{v \in F_i : \text{pos}(v) \geq 1\}, F_i)|.$$

Hence, the desired inequality holds.

Result. Finally, we return yes if and only if $\text{DP}[T, \text{curr}, \text{pos}] \geq t$ for $\text{curr} : F_T \rightarrow \{k\}$ and some function $\text{pos} : F_T \rightarrow \{0, \ell+1\}$. The restriction to these functions is correct, since we can assume without loss of generality that a vertex is active in G_T if and only if its k -th activity interval ends at time step T (recall that we assume that each vertex is in at least $k(\ell+1)+1$ bags). Consequently, all k -activity ℓ -timelines of \mathcal{G} are considered in the definition of the table entry $\text{DP}[T, \text{curr}, \text{pos}]$ and all covered temporal edges of \mathcal{G} are counted in the respective maximum of the definition.

Running Time. For each time step $i \in [T]$, there are $((k+1)(\ell+2))^{\text{vimw}}$ choices for the functions $\text{curr} : F_i \rightarrow [0, k]$ and $\text{pos} : F_i \rightarrow [0, \ell+1]$. This upper-bounds the size of our dynamic programming table by $T \cdot ((k+1)(\ell+2))^{\text{vimw}}$. In each recursive computation of a table entry, there are also $\mathcal{O}(((k+1)(\ell+2))^{\text{vimw}})$ choices for curr' and pos' . The remaining summands of the recursive formula can be computed in $\mathcal{O}(n)$ time. Hence, the overall running time can be bounded by $((k+1)(\ell+2))^{2 \cdot \text{vimw}} \cdot (n+T)^{\mathcal{O}(1)}$. \blacktriangleleft

Now, we show that for TIMELINE PVC the parameter $\text{vimw} = \text{vimw}[1]$ can be replaced by an even smaller parameter, $\text{vimw}[k(\ell+1)+1]$. The algorithm for this parameter has two steps: First it applies a preprocessing step that handles the large bags. Then, it invokes the algorithm of Theorem 4.1 for $\text{vimw} = \text{vimw}[1]$.

► **Proposition 4.2.** *TIMELINE PVC is solvable in $((k+1)(\ell+2))^{4 \cdot \text{vimw}[x]} \cdot (n+T)^{\mathcal{O}(1)}$ time where $x = k(\ell+1) + 1$.*

Proof. Let $(F_i)_{i \in [T]}$ be the bags of the vertex-interval-membership sequence of \mathcal{G} . Similar to the proof of Theorem 4.1, it is safe to assume that each vertex v is contained in at least $k \cdot (\ell+1)$ bags (by definition the bags containing v need to be consecutive) because otherwise, we can greedily pick the k -activity intervals of v such that v is active during all these snapshots and thus all temporal edges incident to v are covered and we can reduce t accordingly.

We call the $k \cdot (\ell+1)$ largest bags of $(F_i)_{i \in [T]}$ *large*. The idea is to pick the intervals greedily for all vertices that are only contained in large bags. Intuitively, this works since the large bags appear consecutively. Afterwards, we remove all these vertices and adapt t accordingly. Finally, we can invoke the algorithm of Theorem 4.1.

Now, let $[i, j] \subset [T]$ be a sequence of consecutive indices such that each bag F_x for each $x \in [i, j]$ is large. By definition $j - i \leq k \cdot (\ell+1) - 1$. Now, let $\mathcal{A} = \bigcup_{x \in [i, j]} F_x \setminus (F_{i-1} \cup F_{j+1})$. If $i = 0$ or $j = T$, then F_{i-1} or F_{j+1} is simply the empty set. Suppose $\mathcal{A} \neq \emptyset$. By definition, each vertex in \mathcal{A} is isolated in G_1, \dots, G_{i-1} and also in G_{j+1}, \dots, G_T . By our assumption that each vertex is contained in at least $k \cdot (\ell+1)$ bags, we can conclude that $j - i = k \cdot (\ell+1) - 1$. Now observe that by picking the k -activity intervals for each vertex in \mathcal{A} greedily, we can cover all temporal edges which are incident to some vertex of \mathcal{A} . Hence, it is safe to remove \mathcal{A} from \mathcal{G} and to reduce t by the number of temporal edges covered by the vertices in \mathcal{A} . For the remaining instance, we use the algorithm of Theorem 4.1. After removing \mathcal{A} , we have $|F_x| \leq |F_{i-1}| + |F_{j+1}| \leq 2 \text{vimw}[k(\ell+1) + 1]$ for every $x \in [i, j]$ and consequently Theorem 4.1 yields the desired running time. Now, suppose $\mathcal{A} = \emptyset$ and let F_x be a large bag contained in a sequence of large bags with indices $[i, j]$. In this case, we can conclude $|F_x| \leq |F_{i-1}| + |F_{j+1}| \leq 2 \text{vimw}[k(\ell+1) + 1]$ and again apply Theorem 4.1. ◀

Now, we focus on TIMELINE (PARTIAL) DOMINATING SET. Initially, we show how to adapt the algorithm of Theorem 4.1 for TIMELINE PDS to obtain an algorithm with the same running time for TIMELINE PDS. However, we need to be more careful since isolated vertices in the snapshots can no longer be ignored as for TIMELINE PVC. This means that vertices which are not contained in the current bag of the vertex-interval-membership sequence might have to be active. Hence, the update of the dynamic program additionally needs to consider the case that activity intervals of vertices appear before or after their first or last occurrence in a bag.

► **Theorem 4.3 (★).** *TIMELINE PDS is solvable in $((k+1)(\ell+2))^{2 \cdot \text{vimw}} \cdot (n+T)^{\mathcal{O}(1)}$ time.*

Recall that for TIMELINE PVC we showed that $\text{vimw} = \text{vimw}[1]$ can be replaced by the much smaller parameter $\text{vimw}[k \cdot (\ell+1) + 1]$, the size of the $k \cdot (\ell+1) + 1$ largest bag. Hence, one may ask whether vimw can be replaced by $\text{vimw}[i]$ with some suitably chosen i for TIMELINE DS and TIMELINE PDS. We show that both problems behave quite differently: First, we show we that vimw can be replaced by $\text{vimw}[\ell+2]$ for TIMELINE DS by having a preprocessing step and then adapting the algorithm behind Theorem 4.3. Second, we show that replacing $\text{vimw} = \text{vimw}[1]$ by $\text{vimw}[2]$ is not possible for TIMELINE PDS. More precisely, we show NP-hardness even if there are only two snapshots one of which is edgeless (which implies $\text{vimw}[2] = 0$).

We start with our result for TIMELINE DS.

► **Proposition 4.4 (★).** *TIMELINE DS is solvable in $((k+1)(\ell+2))^{4 \cdot \text{vimw}[x]} \cdot (n+T)^{\mathcal{O}(1)}$ time where $x = \ell+2$.*

Finally, we show hardness for constant k, ℓ , and $\text{vimw}[2]$ for TIMELINE PDS.

► **Theorem 4.5** (*). *TIMELINE PDS is NP-hard even if $T = 2$, $k = 1$, $\ell = 0$, the underlying graph is bipartite, planar, has maximum degree 3, and one snapshot is edgeless.*

4.2 The Parameter Interval-Membership-Width

Now, we study the influence of imw instead of vimw . Initially, we show that the parameter $\text{imw} + k + \ell$ yields a polynomial kernel for TIMELINE DS. This also implies a polynomial kernel for $\text{vimw} + k + \ell$ and $n + k + \ell$ for TIMELINE DS. More precisely, we show an even stronger result: We provide a polynomial kernel for $q + k + \ell$, where q is the maximal number of edges in any snapshot. Note that q is never larger than imw , since any bag F_i contains E_i . Moreover, in the temporal graph \mathcal{G} with a star on n leaves as underlying graph and lifetime $T = 2n$, where the edge towards the i -th leaf is active at snapshots i and $i + n$, we have $\text{imw} = n$ and $q = 1$.

► **Theorem 4.6.** *TIMELINE DS has a kernel where both the number of snapshots and vertices are cubic in $q + k + \ell$ which can be computed in linear time with respect to the input size.*

Proof. In order to provide the kernel we bound the number of snapshots T and the number n of vertices of the underlying graph based on the following case distinction.

Case 1: $q \geq n/2$. By definition each k -activity ℓ -timeline has active vertices in at most $n \cdot k \cdot (\ell + 1)$ snapshots. Hence, if $T > 2 \cdot q \cdot k \cdot (\ell + 1) \geq n \cdot k \cdot (\ell + 1)$, then the instance is a no-instance.

Case 2.1: $q < n/2$ and $T > 2 \cdot k \cdot (\ell + 1) + 1$. We show that in this case, we are dealing with a no-instance. Since each snapshot contains at most q edges, any set of p vertices can dominate at most $p + q$ vertices in this snapshot. As $q < n/2$, it follows that at least $n/2$ vertices must be active in order to dominate all temporal vertices of the corresponding snapshot G_x . Moreover, note that the total number of active vertices in any k -activity ℓ -timeline is at most $z = n \cdot k \cdot (\ell + 1)$. Consequently, $T \leq z/(n/2) \leq 2 \cdot k \cdot (\ell + 1)$ for any yes-instance.

Case 2.2: $q < n/2$ and $T \leq 2 \cdot k \cdot (\ell + 1) + 1$. If $n \leq 4 \cdot q \cdot k \cdot (\ell + 1)$, then we have the desired bound. Otherwise, if $n \geq 4 \cdot q \cdot k \cdot (\ell + 1) + 1$, we can solve the instance in polynomial time as follows. Since each snapshot has at most q edges, at most $2q$ vertices are non-isolated in each snapshot. Moreover, since there are at most $2 \cdot k \cdot (\ell + 1)$ snapshots, in total at most $4 \cdot q \cdot k \cdot (\ell + 1)$ vertices are non-isolated in the temporal graph. Consequently, since $n \geq 4 \cdot q \cdot k \cdot (\ell + 1) + 1$ there exists at least one vertex v which is isolated in every snapshot. Thus, if $T \geq k \cdot (\ell + 1) + 1$ we deal with a trivial no-instance and if $T \leq k \cdot (\ell + 1)$ we deal with a trivial yes-instance.

Thus, in each case we either solve the instance (giving a kernel of constant size) or obtain a kernel with at most $2 \cdot q \cdot k \cdot (\ell + 1)$ snapshots and at most $4 \cdot q \cdot k \cdot (\ell + 1)$ vertices. Moreover, the kernel can be computed in linear time as it involves only computing n, q , and T . ◀

We show that the remaining problems are NP-hard if $\text{imw} + k + \ell \in \mathcal{O}(1)$.

► **Theorem 4.7** (*). *TIMELINE VC is NP-hard even if $k = 2$, $\ell = 0$, $\text{imw} = 4$, each edge of the underlying graph exists in exactly one snapshot, and the underlying graph has a constant maximum degree.*

► **Theorem 4.8** (*). *TIMELINE PDS is NP-hard even if $k = 1$, $\ell = 0$, $\text{imw} = 36$ and the underlying graph has a constant maximum degree.*

5 Complexity with respect to Input Parameters

Finally, we study the influence of the input parameters n, k, ℓ , and t on the complexity of TIMELINE (PARTIAL) VERTEX COVER and TIMELINE (PARTIAL) DOMINATING SET. Both TIMELINE VC and TIMELINE DS are NP-hard for constant values of k and ℓ . Consequently, larger parameters need to be considered to obtain FPT-algorithms or XP-algorithms. Froese et al. [18, Theorem 8] showed that TIMELINE VC admits an FPT-algorithm for $n + k$. A similar algorithm also works for TIMELINE DS.

► **Theorem 5.1** (\star). *TIMELINE DS is solvable in time $\mathcal{O}(n^{2+nk}T)$.*

Froese et al. [18, Theorem 12] showed that TIMELINE VC is W[1]-hard for n even if $\ell = 1$ with a reduction from UNARY BIN PACKING and using a multicolored and a nonuniform variant of TIMELINE VC as intermediate problems in the reduction. In NONUNIFORM TIMELINE VC we are not given a single number of permitted activity intervals k , but instead a number k_v for each vertex v . In a similar way we introduce NONUNIFORM TIMELINE DS, which we will use as an intermediate problem to show hardness for $n + \ell$ for TIMELINE DS.

► **Theorem 5.2** (\star). *TIMELINE DS parameterized by n is W[1]-hard even if $\ell = 1$.*

The W[1]-hardness results for both problems do not apply to the case $\ell = 0$. For TIMELINE VC, this case is fixed-parameter tractable because there exists an ILP formulation in which the number of variables is bounded by some function of n [18, Lemma 10]. It is also straightforward to extend this ILP formulation to TIMELINE PVC. To complete the picture, we extend the ILP formulation also to TIMELINE PDS, thus showing that it is fixed-parameter tractable for $\ell = 0$ as well.

► **Theorem 5.3** (\star). *TIMELINE PDS is fixed-parameter tractable with respect to n if $\ell = 0$.*

Finally, we show that both TIMELINE PDS and TIMELINE PVC can be solved in $2^{\mathcal{O}(t)} \cdot (n+T)^{\mathcal{O}(1)}$ time, where t denotes the number of temporal vertices/edges, which need to be dominated/covered. Recall that Dondi et al. [12, Theorem 6] provided an FPT-algorithm for TIMELINE PVC with running time $t^t \cdot (n+T)^{\mathcal{O}(1)}$ if $k = 1$. Hence, our approach improves upon their running time and is not limited to $k = 1$. The idea is to use color coding [2], a very popular technique to obtain FPT-algorithms [7]. To provide deterministic algorithm, we use (n, k) -perfect hash families which can be computed efficiently [26].

► **Theorem 5.4** (\star). *TIMELINE PDS and TIMELINE PVC are solvable in $2^{\mathcal{O}(t)} \cdot (n+T)^{\mathcal{O}(1)}$ time.*

6 Conclusion

We studied the classical and parameterized complexity of TIMELINE (PARTIAL) VERTEX COVER and TIMELINE (PARTIAL) DOMINATING SET. We showed that all problems admit FPT-algorithms for $\text{vimw} + k + \ell$, where vimw is the vertex-interval-membership width. Our running time bounds also give XP-algorithms for vimw alone. For TIMELINE VC this improves upon an XP-algorithm for n [18]. Moreover, we showed that the smaller parameter $\text{imw} + k + \ell$ leads to para-NP-hardness for all problems except TIMELINE DS. Hence, we discovered a parameter where a dominating set problem is tractable while the corresponding vertex cover variant is not.

For future work it is interesting to investigate whether already the parameter $\text{vimw} + k$ yields an FPT-algorithm for any problem in our study. Moreover, it is open whether TIMELINE PVC and TIMELINE PDS admit an FPT-algorithm with respect to $n + k$. Even for $k = 1$ fixed-parameter tractability with respect to n is still open. Moreover, one could investigate the complexity with respect to other temporal graph parameters such as the temporal neighborhood diversity [15] or temporal graph parameters that are sensitive to the order of the snapshots [5, 6].

A further problem related to TIMELINE VC is MINTIMELINE_+ . In this problem, the value of ℓ bounds the sum of the lengths of the activity intervals instead of the length of the longest activity interval. MINTIMELINE_+ was also introduced by Rozhenstein et al. [28] and studied in several works [9–11, 13, 18, 24, 29]. To distinguish this problem from the other problems in our naming convention, TIMELINE VC ($\text{MINTIMELINE}_\infty$) could be called MINMAX TIMELINE VC and MINTIMELINE_+ could be called SUM TIMELINE VC, and the timeline variants of DOMINATING SET could be named analogously. It is open which of our positive results for vimw and imw can be transferred to the SUM variants of the problems. Finally, it is interesting to study other classic problems as FEEDBACK VERTEX SET in the MINMAX TIMELINE and SUM TIMELINE setting.

References

- 1 Eleni C. Akrida, George B. Mertzios, Paul G. Spirakis, and Viktor Zamaraev. Temporal vertex cover with a sliding time window. *J. Comput. Syst. Sci.*, 107:108–123, 2020. doi:10.1016/J.JCSS.2019.08.002.
- 2 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.
- 3 Benjamin Merlin Bumpus and Kitty Meeks. Edge exploration of temporal graphs. *Algorithmica*, 85(3):688–716, 2023. doi:10.1007/S00453-022-01018-7.
- 4 Arnaud Casteigts. *A Journey through Dynamic Networks (with Excursions)*. Habilitation thesis, Université de Bordeaux, 2018. URL: <https://tel.archives-ouvertes.fr/tel-01883384>.
- 5 Arnaud Casteigts, Nils Morawietz, and Petra Wolf. Distance to Transitivity: New Parameters for Taming Reachability in Temporal Graphs. In *Proceedings of the 49th International Symposium on Mathematical Foundations of Computer Science (MFCS '24)*, volume 306 of *LIPICs*, pages 36:1–36:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.MFCS.2024.36.
- 6 Filippos Christodoulou, Pierluigi Crescenzi, Andrea Marino, Ana Silva, and Dimitrios M. Thilikos. Making the interval membership width of temporal graphs connected and bidirectional. In *Proceedings of the 35th International Workshop on Combinatorial Algorithms (IWOCA '24)*, volume 14764 of *Lecture Notes in Computer Science*, pages 247–258. Springer, 2024. doi:10.1007/978-3-031-63021-7_19.
- 7 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 8 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate Texts in Mathematics*. Springer, 2012.
- 9 Riccardo Dondi. Insights into the Complexity of Disentangling Temporal Graphs. In *Proceedings of the 23rd Italian Conference on Theoretical Computer Science (ICTCS '22)*, volume 3284 of *CEUR Workshop Proceedings*, pages 1–13. CEUR-WS.org, 2022. URL: <https://ceur-ws.org/Vol-3284/2973.pdf>.
- 10 Riccardo Dondi. Untangling temporal graphs of bounded degree. *Theor. Comput. Sci.*, 969:114040, 2023. doi:10.1016/J.TCS.2023.114040.

- 11 Riccardo Dondi and Manuel Lafond. An FPT algorithm for timeline cover. *J. Comput. Syst. Sci.*, 154:103679, 2025. doi:10.1016/J.JCSS.2025.103679.
- 12 Riccardo Dondi, Fabrizio Montecchiani, Giacomo Ortali, Tommaso Piselli, and Alessandra Tappini. Partial temporal vertex cover with bounded activity intervals. In *Proceedings of the 3rd Symposium on Algorithmic Foundations of Dynamic Networks (SAND '24)*, volume 292 of *LIPICs*, pages 11:1–11:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.SAND.2024.11.
- 13 Riccardo Dondi and Alexandru Popa. Exact and approximation algorithms for covering timeline in temporal graphs. *Ann. Oper. Res.*, 351(1):609–628, 2025. doi:10.1007/s10479-024-05993-8.
- 14 Rodney G. Downey and Michael Ralph Fellows. *Fundamentals of Parameterized Complexity*. Springer Science & Business Media, 2013.
- 15 Jessica A. Enright, Samuel D. Hand, Laura Larios-Jones, and Kitty Meeks. Structural parameters for dense temporal graphs. In *Proceedings of the 49th International Symposium on Mathematical Foundations of Computer Science (MFCS '24)*, volume 306 of *LIPICs*, pages 52:1–52:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.MFCS.2024.52.
- 16 David Eppstein and Emma S. Spiro. The h-index of a graph and its application to dynamic subgraph statistics. *J. Graph Algorithms Appl.*, 16(2):543–567, 2012. doi:10.7155/JGAA.00273.
- 17 Till Fluschnik, Rolf Niedermeier, Valentin Rohm, and Philipp Zschoche. Multistage vertex cover. *Theory Comput. Syst.*, 66(2):454–483, 2022. doi:10.1007/s00224-022-10069-w.
- 18 Vincent Froese, Pascal Kunz, and Philipp Zschoche. Disentangling the computational complexity of network untangling. *Theory Comput. Syst.*, 68(1):103–121, 2024. doi:10.1007/S00224-023-10150-Y.
- 19 M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete problems. In *Proceedings of the 6th Annual ACM Symposium on Theory of Computing*, pages 47–63. ACM, 1974. doi:10.1145/800119.803884.
- 20 Thekla Hamm, Nina Klobas, George B. Mertzios, and Paul G. Spirakis. The complexity of temporal vertex cover in small-degree graphs. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI '22)*, pages 10193–10201. AAAI Press, 2022. doi:10.1609/AAAI.V36I9.21259.
- 21 Anton Herrmann. On the complexity of computing optimal dominating sets in temporal graphs. Master's thesis, Friedrich-Schiller-Universität Jena, 2024.
- 22 Anton Herrmann, Christian Komusiewicz, Nils Morawietz, and Frank Sommer. Temporal dominating set and temporal vertex cover under the lense of degree restrictions. In *Proceedings of the 4th Symposium on Algorithmic Foundations of Dynamic Networks (SAND '25)*, volume 330 of *LIPICs*, pages 16:1–16:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPICS.SAND.2025.16.
- 23 David C. Kutner and Laura Larios-Jones. Temporal reachability dominating sets: Contagion in temporal graphs. *J. Comput. Syst. Sci.*, 155:103701, 2026. doi:10.1016/J.JCSS.2025.103701.
- 24 Giorgio Lazzarinetti, Sara Manzoni, Italo Zoppis, and Riccardo Dondi. FastMinTC+: A Fast and Effective Heuristic for Minimum Timeline Cover on Temporal Networks. In *Proceedings of the 31st International Symposium on Temporal Representation and Reasoning (TIME '24)*, volume 318 of *LIPICs*, pages 20:1–20:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.TIME.2024.20.
- 25 Jayadev Misra and David Gries. A constructive proof of Vizing's theorem. *Inf. Process. Lett.*, 41(3):131–133, 1992. doi:10.1016/0020-0190(92)90041-S.
- 26 Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS '95)*, pages 182–191. IEEE Computer Society, 1995. doi:10.1109/SFCS.1995.492475.

- 27 Tayler Pino, Salimur Choudhury, and Fadi Al-Turjman. Dominating set algorithms for wireless sensor networks survivability. *IEEE Access*, 6:17527–17532, 2018. doi:10.1109/ACCESS.2018.2819083.
- 28 Polina Rozenshtein, Nikolaj Tatti, and Aristides Gionis. The network-untangling problem: from interactions to activity timelines. *Data Min. Knowl. Discov.*, 35(1):213–247, 2021. doi:10.1007/S10618-020-00717-5.
- 29 Carsten Schubert. Leveraging graph structure to untangle temporal networks efficiently. Master’s thesis, TU Berlin, Institute of Software Engineering and Theoretical Computer Science, 2023.
- 30 I. Stojmenovic, M. Seddigh, and J. Zunic. Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEE Trans. Parallel Distributed Syst.*, 13(1):14–25, 2002. doi:10.1109/71.980024.
- 31 M Verheije. Algorithms for domination problems on temporal graphs. Master’s thesis, Utrecht University, Graduate School of Natural Sciences, 2021.
- 32 Vadim G Vizing. On an estimate of the chromatic class of a p -graph. *Discret Analiz*, 3:25–30, 1964.