The Complexity of Temporal Vertex Cover in Small-Degree Graphs

Thekla Hamm[1], Nina Klobas[2], George B. Mertzios[2*], Paul G. Spirakis[3†]

[1] Algorithms and Complexity Group, TU Wien, Austria
[2] Department of Computer Science, Durham University, UK
[3] Department of Computer Science, University of Liverpool, UK
[3] Computer Engineering & Informatics Department, University of Patras, Greece
thamm@ac.tuwien.ac.at, {nina.klobas, george.mertzios}@durham.ac.uk, p.spirakis@liverpool.ac.uk

Abstract

Temporal graphs naturally model graphs whose underlying topology changes over time. Recently, the problems Temporal Vertex Cover (or TVC) and Sliding-Window Temporal Vertex Cover (or Δ-TVC for time-windows of a fixed-length Δ) have been established as natural extensions of the classic Vertex Cover problem on static graphs with connections to areas such as surveillance in sensor networks.

In this paper we initiate a systematic study of the complexity of TVC and Δ-TVC on sparse graphs. Our main result shows that for every $\Delta \geq 2$, Δ-TVC is NP-hard even when the underlying topology is described by a path or a cycle. This resolves an open problem from literature and shows a surprising contrast between Δ-TVC and TVC for which we provide a polynomial-time algorithm in the same setting. To circumvent this hardness, we present a number of exact and approximation algorithms for temporal graphs whose underlying topologies are given by a path, that have bounded vertex degree in every time step, or that admit a small-sized temporal vertex cover.

1 Introduction

A great variety of modern, as well as of traditional networks, are dynamic in nature as their link availability changes over time. Information and communication networks, social networks, transportation networks, and various physical systems are only a few indicative examples of such inherently dynamic networks (Holme and Saramäki 2013; Michail and Spirakis 2018). All these application areas share the common characteristic that the network structure, i.e. the underlying graph topology, is subject to *discrete changes over time*. In this paper, embarking from the foundational work of Kempe et al. (Kempe, Kleinberg, and Kumar 2002), we adopt the following simple and natural model for time-varying networks, which is given by a graph with sets of time-labels associated with its edges, while the vertex set is fixed.

Definition 1 (Temporal Graph)**.** A *temporal graph* is a pair (G, λ), where $G = (V, E)$ is an underlying (static) graph and $\lambda : E \to 2^{\mathbb{N}}$ is a *time-labeling* function which assigns to every edge of G a set of discrete-time labels.

For an edge $e \in E$ in the underlying graph G of a temporal graph (G, λ), if $t \in \lambda(e)$ then we also say that e is *active* at time t in (G, λ). That is, for every edge $e \in E$, $\lambda(e)$ denotes the set of time slots at which e is *active*. Due to their relevance and applicability in many areas, temporal graphs have been studied from various perspectives and under different names such as *dynamic* (Giakkoupis, Sauerwald, and Stauffer 2014; Casteigts et al. 2012), *evolving* (Bui-Xuan, Ferreira, and Jarry 2003; Ferreira 2004; Clementi et al. 2010), *time-varying* (Flocchini, Mans, and Santoro 2009; Tang et al. 2010; Aaron, Krizanc, and Meyerson 2014), and *graphs over time* (Leskovec, Kleinberg, and Faloutsos 2007). For a comprehensive overview on the existing models and results on temporal graphs from a distributed computing perspective see the surveys (Casteigts et al. 2012; Casteigts and Flocchini 2013a,b).

Mainly motivated by the fact that, due to causality, information can be transfered in a temporal graph along sequences of edges whose time-labels are increasing, the most traditional research on temporal graphs has focused on temporal paths and other "path-related" notions, such as e.g. temporal analogues of distance, reachability, exploration and centrality (Klobas et al. 2021; Heeger et al. 2021; Akrida et al. 2016; Erlebach, Hoffmann, and Kammer 2021; Mertzios, Michail, and Spirakis 2019; Michail and Spirakis 2016; Akrida et al. 2017; Enright et al. 2021; Zschoche et al. 2020; Casteigts et al. 2021). To complement this direction, several attempts have been recently made to define meaningful "non-path" temporal graph problems which appropriately model specific applications. Motivated by the contact patterns among high-school students, Viard et al. (Viard, Latapy, and Magnien 2016), introduced Δ-cliques, an extension of the concept of cliques to temporal graphs (see also (Himmel et al. 2017; Bentert et al.

*Supported by the EPSRC grant EP/P020372/1.
†Supported by the NeST initiative of the School of EEE and CS at the University of Liverpool and by the EPSRC grant EP/P02002X/1.

2018)). Chen et al. (Chen et al. 2018) presented an extension of the cluster editing problem to temporal graphs, in which all vertices interact with each other at least once every Δ consecutive time steps within a given time interval. Furthermore, Akrida et al. (Akrida et al. 2020) introduced the notion of temporal vertex cover (also with a sliding time window), motivated by applications of covering problems in sensor networks. Further examples of meaningful "non-path" temporal graph problems include variations of temporal graph coloring (Mertzios, Molter, and Zamaraev 2021; Yu et al. 2013; Ghosal and Ghosh 2015) in the context of planning and channel assignment in mobile sensor networks, and the temporally transitive orientations of temporal graphs (Mertzios et al. 2021).

The problems Temporal Vertex Cover (or TVC) and Sliding-Window Temporal Vertex Cover (or Δ-TVC for time-windows of a fixed-length Δ) have been established as natural extensions of the well-known Vertex Cover problem on static graphs (Akrida et al. 2020). Given a temporal graph $\mathcal{G}$, the aim of TVC is to cover every edge at least once during the lifetime T of $\mathcal{G}$, where an edge can be covered by one of its endpoints, and only at a time step when it is active. For any $\Delta \in \mathbb{N}$, the aim of the more "pragmatic" problem Δ-TVC is to cover every edge at least once at every Δ consecutive time steps. In both problems, we try to minimize the number of "vertex appearances" in the resulting cover, where a vertex appearance is a pair (v, t) for some vertex v and $t \in \{1, 2, \ldots, T\}$.

TVC and Δ-TVC naturally generalize the applications of the static problem Vertex Cover to more dynamic inputs, especially in the areas of wireless ad hoc networks, as well as network security and scheduling. In the case of a static graph, the vertex cover can contain trusted vertices which have the ability to monitor/surveil all transmissions (Ileri et al. 2016; Richter, Helmert, and Gretton 2007) or all link failures (Kavalci, Ural, and Dagdeviren 2014) between any pair of vertices through the edges of the graph. In the temporal setting, it makes sense to monitor the transmissions and to check for link failures within every sliding time window of an appropriate length Δ (which is exactly modeled by Δ-TVC).

It is already known that both TVC and Δ-TVC are NP-hard; for Δ-TVC this is even the case when $\Delta = 2$ and the minimum degree of the underlying graph G is just 3 (Akrida et al. 2020). One of the most intriguing questions left open (see Problem 1 in (Akrida et al. 2020)) is whether Δ-TVC (or, more generally, Sliding-Window Temporal Vertex Cover) can be solved in polynomial time.

Our Contribution. In this paper we initiate the study of the complexity of TVC and Δ-TVC on sparse graphs. Our main result (see Section 3.1) is that, for every $\Delta \geq 2$, Δ-TVC is NP-hard even when G is a path or a cycle, while TVC can be solved in polynomial time on paths and cycles. This resolves the first open question (Problem 1) of (Akrida et al. 2020). In contrast, we show that TVC (see Section 3.2) can be solved in polynomial time on temporal paths and cycles. Moreover, for any $\Delta \geq 2$, we provide a *Polynomial-Time Approximation Scheme (PTAS)* for Δ-TVC on temporal paths and cycles (see Section 3.2), which also complements our hardness result for paths.

The NP-hardness of Section 3.1 signifies that an optimum solution for Δ-TVC is hard to compute, even for $\Delta = 2$ and under severe degree restrictions of the input instance. To counter this hardness for more general temporal graphs than those with underlying paths and cycles as in Section 3, in Section 4 we give three algorithms for every $\Delta \geq 2$: First we present an exact algorithm for Δ-TVC with single exponential running time dependency on the number of edges in the underlying graph (see Section 4.1). Using this algorithm we are able to devise for any $d \geq 3$ a polynomial-time $(d-1)$-approximation (see Section 4.2), where d is the maximum vertex degree in any time step, i. e., in any part of the temporal graph that is active at the same time. This improves the currently best known d-approximation algorithm for Δ-TVC (Akrida et al. 2020) and thus also answers another open question (Problem 2 in (Akrida et al. 2020)). Finally, we present a simple fixed-parameter tractable algorithm with respect to the size of an optimum solution (see Section 4.3).

2 Preliminaries

Given a (static) graph $G = (V, E)$ with vertices in V and edges in E, an edge between two vertices u and v is denoted by uv, and in this case u and v are said to be *adjacent* in G. For every $i, j \in \mathbb{N}$, where $i \leq j$, we let $[i, j] = \{i, i+1, \ldots, j\}$ and $[j] = [1, j]$. Throughout the paper we consider temporal graphs whose underlying graphs are finite and whose time-labeling functions only map to finite sets. This implies that there is some $t \in \mathbb{N}$ such that, for every $t' > t$, no edge of G is active at t' in (G, λ). We denote the smallest such t by T, i. e., $T = \max\{t \in \lambda(e) \mid e \in E\}$, and call T the *lifetime* of (G, λ). Unless otherwise specified, n denotes the number of vertices in the underlying graph G, and T denotes the lifetime of the temporal graph $\mathcal{G}$. We refer to each integer $t \in [T]$ as a *time slot* of (G, λ). The *instance* (or *snapshot*) of (G, λ) *at time* t is the static graph $G_t = (V, E_t)$, where $E_t = \{e \in E : t \in \lambda(e)\}$.

A *temporal path* of length k is a temporal graph $\mathcal{P} = (P, \lambda)$, where the underlying graph P is the path $(v_0, v_1, v_2, \ldots, v_k)$ on $k+1$ vertices, with edges $e_i = v_{i-1}v_i$ for $i = 1, 2, \ldots, k$. In many places throughout the paper, we visualize a temporal path as a 2-dimensional array $V(P) \times [T]$, where two vertices $(x, t), (y, t') \in V(P) \times [T]$ are connected in this array if and only if $t = t' \in \lambda(xy)$ and $xy \in E(P)$. For example see Figure 1.

For every $t = 1, \ldots, T - \Delta + 1$, let $W_t = [t, t + \Delta - 1]$ be the Δ-time window that starts at time t. For every $v \in V$ and every time slot t, we denote the *appearance*

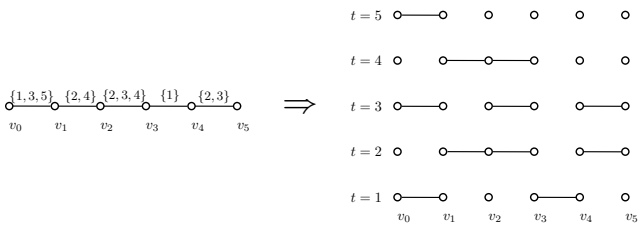


Figure 1: An example of visualizing a temporal path graph $\mathcal{G}$ as a 2-dimensional array, in which every edge corresponds to a time-edge of $\mathcal{G}$.

of vertex v at time t by the pair (v,t) and the *edge appearance* (or *time-edge*) of e at time t by (e,t).

A *temporal vertex subset* of (G,λ) is a set of vertex appearances in (G,λ), i.e. a set of the form $S \subseteq \{(v,t) \mid v \in V, t \in [T]\}$. For a temporal vertex subset S and some Δ-time window W_i within the lifetime of (G,λ), we denote by $S[W_i] = \{(v,t) \in S \mid t \in W_i\}$ the subset of all vertex appearances in S in the Δ-time window W_i. For a Δ-time window W_i within the lifetime of a temporal graph (G,λ), we denote by $E[W_i] = \{e \in E \mid \lambda(e) \cap W_i \neq \emptyset\}$ the set of all edges which appear at some time slot within W_i.

A temporal vertex subset $\mathcal{C}$ is a *sliding Δ-time window temporal vertex cover*, or Δ-TVC for short, of a temporal graph (G,λ) if, for every Δ-time window W_i within the lifetime of (G,λ) and for every edge in $E[W_i]$, there is some $(v,t) \in \mathcal{C}[W_i]$ such that $v \in e$, i.e. v is an endpoint of e, and $t \in \lambda(e)$. Here we also say (v,t) *covers* (e,t) in time window W_i.

3 Paths and Cycles

In Section 3.1 we provide our main NP-hardness result for Δ-TVC, for any $\Delta \geq 2$, on instances whose underlying graph is a path or a cycle (see Theorem 3 and Corollary 4). In Section 3.2 we prove that TVC on underlying paths and cycles is polynomially solvable, and we also provide our PTAS for Δ-TVC on underlying paths and cycles, for every $\Delta \geq 2$.

3.1 Hardness on Temporal Paths and Cycles

Our NP-hardness reduction of Theorem 3 is done from the NP-hard problem *planar monotone rectilinear* 3 *satisfiability* (or *planar monotone rectilinear 3SAT*), see (de Berg and Khosravi 2010). This is a specialization of the classical Boolean 3-satisfiability problem to a planar incidence graph. A Boolean satisfiability formula ϕ in conjunctive normal form (CNF) is called *monotone*, if each clause of ϕ consists of only positive or only negative literals. We refer to these clauses as *positive* and *negative* clauses, respectively. By possibly repeating literals, we may assume without loss of generality that every clause contains exactly three (not necessarily different) literals.

In an instance of planar monotone rectilinear 3SAT, each variable and each clause is represented with a horizontal line segment, as follows. The line segments of all variables lie on the same horizontal line on the plane, which we call the *variable-axis*. For every clause $C = (x_i \vee x_j \vee x_k)$ (or $C = (\overline{x_i} \vee \overline{x_j} \vee \overline{x_k})$), the line segment of C is connected via straight vertical line segments to the line segments of x_i, x_j and of x_k, such that every two (horizontal or vertical) line segments are pairwise non-intersecting. Furthermore, every line segment of a positive (resp. negative) clause lies above (resp. below) the variable-axis. Finally, by possibly slightly moving the clause line segments higher or lower, we can assume without loss of generality that every clause line segment lies on a different horizontal line on the plane.

Let ϕ be an arbitrary instance of planar monotone rectilinear 3SAT, where $X = \{x_1, \ldots, x_n\}$ is its set of Boolean variables and $\phi(X) = \{C_1, \ldots, C_m\}$ is its set of clauses. We construct from ϕ a temporal path $\mathcal{G}_\phi$ and prove that there exists a truth assignment of X which satisfies $\phi(X)$ if and only if the optimum value of 2-TVC on $\mathcal{G}_\phi$ is at most $f(\mathcal{G}_\phi)$. The exact value of $f(\mathcal{G}_\phi)$ will be defined later.

High-Level Description Given a representation (i.e. embedding) R_ϕ of an instance ϕ of planar monotone rectilinear 3SAT, we construct a 2-dimensional array of the temporal path $\mathcal{G}_\phi$, where:

- every vertical line segment in R_ϕ is associated with an edge of $\mathcal{G}_\phi$ that appears in consecutive time steps,
- every variable (horizontal) line segment in R_ϕ is associated with one or more *segment blocks* (to be formally defined below) in $\mathcal{G}_\phi$, and
- every clause (horizontal) line segment in R_ϕ, corresponding to the clause $C = (x_i \vee x_j \vee x_k)$ (resp. $C = (\overline{x_i} \vee \overline{x_j} \vee \overline{x_k})$), is associated with a clause gadget in $\mathcal{G}_\phi$, which consists of three segment blocks (one for each of x_i, x_j, x_k) together with a path connecting them in the 2-dimensional array for $\mathcal{G}$ (we call this path the *clause gadget connector*).

The exact description of the variables' and clauses' gadgets is given below; first we need to precisely define the segment blocks.

Segment Blocks are used to represent variables. Every segment block consists of a path of length 7 on vertices $(u_0, u_1, \ldots, u_7)$, where the first and last edges (i. e., u_0u_1 and u_6u_7) appear at 9 consecutive time steps starting at time t and ending at time $t+8$, with all other edges appearing only two times, i. e., at times $t+1$ and $t+7$.

Time-edges which correspond to the first and last appearances of u_0u_1 and u_6u_7 in a segment block are called *dummy time-edges*, all remaining time-edges form two (bottom and top) horizontal paths, and two (left and right) vertical sequences of time-edges (which we call here *vertical paths*), . Using the next technical lemma will allow us to model the two different truth values of each variable x_i (True, resp. False) via two different optimum solutions of 2-TVC on a segment block

(namely the "orange and green", resp. "orange and red" temporal vertex covers of the segment block, see Figure 2).

Lemma 2. *There are exactly two different optimum solutions for 2-TVC of a segment block, both of size 15.*

Proof. Let $\mathcal{C}$ be a 2-TVC of a segment block on vertices $u_0, \ldots, u_7$ that starts at time t and finishes at time $t+8$.

In order to cover the dummy time-edges in time windows W_{t-1} and W_{t+8} one of their endpoints has to be in $\mathcal{C}$. Now let us start with the covering of the first edge (u_0u_1) at time $t+1$. Since the dummy time-edges are covered, the edge u_0u_1 is covered in the time window W_t but it is not yet covered in the time window W_{t+1}. We have two options, either cover it at time $t+1$ or $t+2$.

- Suppose that we cover the edge u_0u_1 at $t+1$, then the next time step it has to be covered is $t+3$, the next one $t+5$ and the last one $t+7$. Now that the left vertical path is covered we proceed to cover the bottom and top horizontal paths. The middle 5 edges, from u_1 to u_6, appear only at time steps $t+1$ and $t+7$. Since we covered the edge u_0u_1 at time $t+1$, we can argue that the optimum solution includes the temporal vertex $(u_1, t+1)$ and therefore the edge u_1u_2 is also covered. Extending this covering optimally to the whole path we need to add every second vertex to $\mathcal{C}$, i. e., vertex appearances $(u_3, t+1)$ and $(u_5, t+1)$. Similarly it holds for the vertex appearances of vertices $u_1, \ldots, u_6$ at time $t+7$. The last thing we need to cover is the right vertical path. Since the edge u_6u_7 is covered at time t, the next time step we have to cover it is $t+2$, which forces the next cover to be at $t+4$ and last one at $t+6$. In total $\mathcal{C}$ consists of 4 endpoints of the dummy time-edges, 4 vertices of the left and 3 of the right vertical path, 2 vertices of the bottom and 2 of the top horizontal path. All together we used 11 vertices to cover vertical and horizontal paths and 4 for dummy time-edges. The above described 2-TVC corresponds to the red coloring of the odd segment block depicted in the Figure 2 (left).
  Let us also emphasize that, with the exception of times $t+1$ and $t+7$, we do not distinguish between the solutions that uses a different endpoint to cover the first and last edge. For example if a solution covers the edge u_0u_1 at time $t+2$ then we do not care which of $(u_0, t+2)$ or $(u_1, t+2)$ is in the TVC.
- Covering the edge u_0u_1 at time $t+2$ produces the 2-TVC that is a mirror version of the previous one on the vertical and horizontal paths. More precisely, in this case the covering consists of 3 vertices of the left and 4 of the right vertical path and again 2 vertices of the bottom and 2 of the top horizontal path, together with 4 vertices covering the dummy time-edges.
  This 2-TVC corresponds to the green coloring of the odd segment block depicted in the Figure 2.

Starting with vertex appearances from one optimum solution adding vertex appearances from the other optimum solution either creates a 2-TVC of bigger size or leaves some edges uncovered. □

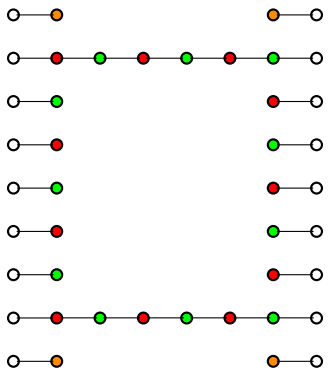


Figure 2: An example of two optimum covers (red and green) of a segment block, where the orange vertices are always included in the solution.

For each variable x_i we create multiple copies of segment blocks, and some specific pairs of these segment blocks are connected to each other via the so-called "horizontal bridges" or "vertical bridges". Two segment blocks, which are connected via a horizontal bridge, start at the same time t but are built on different sets of vertices (i. e., one is to the left of the other in the 2-dimensional array) Similarly, two segment blocks, which are connected via a vertical bridge, are built on the same set of vertices but in different time steps (i. e., one is above the other in the 2-dimensional array).

All the copies have to be created in such a way, that their optimum 2-TVCs depend on each other. In the following we describe two ways to connect two different segment blocks (both for the same variable x_i). As we prove, there are two ways to optimally cover these constructions: one using the "orange and green", and one using the "orange and red" temporal vertex appearances (thus modeling the truth values True and False of variable x_i in our reduction).

As, for every $\Delta \geq 2$, there is a known polynomial-time reduction from Δ-TVC to $(\Delta+1)$-TVC (Akrida et al. 2020), we obtain the following.

Theorem 3. *For every $\Delta \geq 2$, Δ-TVC on instances on an underlying path is NP-hard.*

With a slight modification to the $\mathcal{G}_\phi$ we can create the temporal cycle from R_ϕ and therefore the following holds.

Corollary 4. *For every $\Delta \geq 2$, Δ-TVC on instances on an underlying cycle is NP-hard.*

Proof. We follow the same procedure as above where we add one extra vertex w, to the underlying graph P of $\mathcal{G}_\phi$. We add also two time-edges connecting the first and the last vertex of the temporal path graph $\mathcal{G}_\phi$ at time 1. This increases the size of the 2-TVC by 1 (as we need to include the vertex appearance $(w, 1)$) and it transforms the underlying path P into a cycle. □

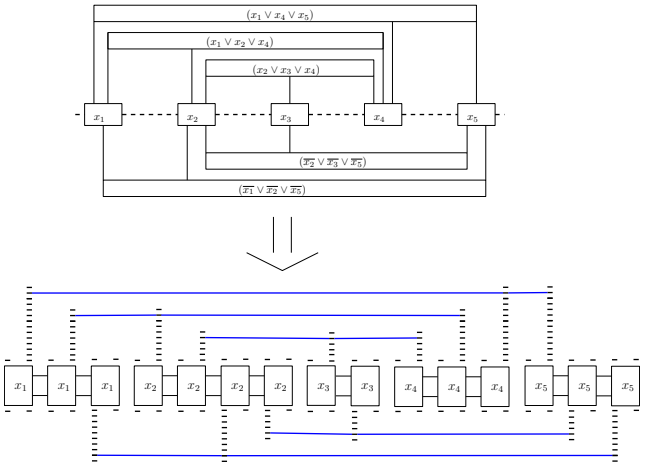


Figure 3: An example of the construction of temporal graph from a planar rectilinear embedding of monotone 3SAT.

3.2 Algorithmic Results

To complement the hardness presented in Section 3.1, we present two polynomial-time algorithms: Firstly, a dynamic program for solving TVC on instances with underlying paths and cycles shows that the hardness is inherently linked to the sliding time windows. Secondly, we give a PTAS for Δ-TVC on instances with underlying paths. This approximation scheme can be obtained using a powerful general purpose result commonly used for approximating geometric problems (Mustafa and Ray 2010).

Theorem 5. *TVC can be solved in polynomial time on instances with a path/cycle as their underlying graph.*

Next we turn to approximating Δ-TVC on underlying paths. The Geometric Hitting Set problem takes as instance a set of geometric objects, e.g. shapes in the plane, and a set of points. The task is to find the smallest set of points, that hit all of the objects. In the paper by Mustafa and Ray (Mustafa and Ray 2010) the authors present a PTAS for the problem, when the geometrical objects are r-admissible set regions. We transform an arbitrary temporal path to the setting of the geometric hitting set. As a result, we obtain a PTAS for the Δ-TVC problem.

Theorem 6. *For every $\varepsilon > 0$, there exists an $(1+\varepsilon)$-approximation algorithm for Δ-TVC on instances with a path as their underlying graph, which runs in time*

$$O\left(n(T-\Delta+1)\cdot(T(n+1))^{\mathcal{O}(\varepsilon^{-2})}\right) = O\left((T(n+1))^{\mathcal{O}(\varepsilon^{-2})}\right),$$

i. e., the problem admits a PTAS.

Proof. Let $\mathcal{G} = (G, \lambda)$ be a temporal path, on vertices $\{v_1, v_2, \ldots, v_n\}$, with lifetime T. We first have to create the range space $R = (P, D)$, where $P \subseteq V \times [T]$ is a set of vertex appearances and D is a set of 2-admissible regions.

Set P of time vertices consist of vertex appearances (v_i, t) for which $t \in \lambda(e_i) \cup \lambda(e_{i+1})$. Intuitively, if edges incident to v do not appear at time t, then (v, t) is not in P. Set D of 2-admissible regions consists of rectangles of 2 different sizes. For every edge e_i that appears in the time window W_t we create one rectangle R_i^t, that includes all vertex appearances incident to e_i in W_t. For example, if edge e_i appears in the time window W_t at times t_1 and t_2, then the corresponding rectangle R_i^t contains vertex appearances $(v_{i-1}, t_1), (v_i, t_1), (v_{i-1}, t_2)$ and (v_i, t_2).

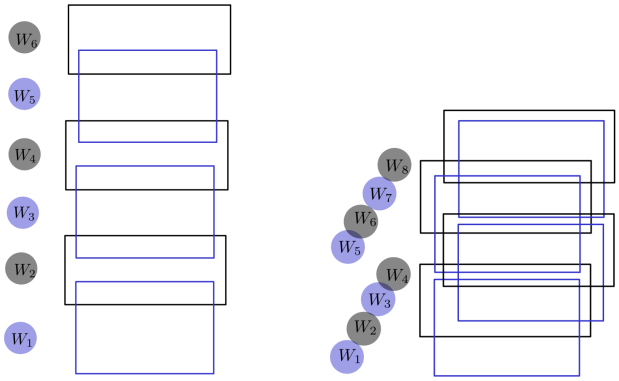
It is not hard to see that $|P| \leq |V| \cdot T = (n+1)T$ and $|D| \leq |E|(T - \Delta + 1) = n(T - \Delta + 1)$.

Formal Construction. Since D will be defined to be a set of 2-admissible regions, the boundary of any two rectangles we construct should intersect at most 2 times. For this purpose we use rectangles, of two different sizes. Let us denote with A the rectangle of size $a_1 \times a_2$ and with B be the rectangle of size $b_1 \times b_2$, where $a_1 > b_1 > b_2 > a_2$.

As we said, for every edge e_i that appears in the time window W_t we construct exactly one rectangle. These are the rules we use to correctly construct them.

1. For a fixed time window W_t we construct the rectangles in such a way, that they intersect only in the case when their corresponding edges e_i, e_j share the same endpoint in the underlying graph G. Since G is a path, the intersection happens only in the case when $j = i + 1$. We can observe also, that there are no three (or more) edges sharing the same endpoint and therefore no three rectangles intersect.
   We require also, that the rectangles corresponding to a pair of the adjacent edges are not the same, i. e., they alternate between form A and B.
2. For any edge e_i, rectangles corresponding to two consecutive time windows W_t, W_{t+1} are not the same, i. e., they alternate between the form A and B. For an example see Figure 4a.
   When the time windows are of size Δ, there are at most Δ rectangles intersecting at every time step t. This holds, because if the edge e_i appears at time t it is a part of the time windows $W_{t_\Delta+1}, W_{t_\Delta+2}, \ldots, W_t$. Since the constructed rectangles are of two sizes, if $\Delta \geq 3$ we create intersections with infinite number of points between the boundary of some rectangles, if we just "stack" the rectangles upon each other. Therefore, we need to shift (in the horizontal direction) rectangles of the same form in one Δ time window. Since the time window W_t never intersects with $W_{t+\Delta}$, we can shift the time windows $W_{t+1}, \ldots, W_{t+\Delta-1}$ and fix the $W_{t+\Delta}$ at the same horizontal position as W_t . For an example see Figure 4b.

Combining both of the above rules we get a grid of rectangles. Moving along the x axis corresponds to mov-



(a) Alternating regions for an edge.

(b) Regions for an edge with shift, when $\Delta = 4$. Each region W_i starts at same horizontal position as W_{i-4}.

Figure 4: Creating regions for one edge.

ing through the edges of the path and moving along the y axis corresponds to moving through the time steps.

By construction, rectangles alternate between the form A and B in both dimensions. If an edge e_i does not appear in the time window W_t, then we do not construct the corresponding rectangle R_i^t. The absence of a rectangle from a grid does not change the pattern of others rectangles. To determine of what form a rectangle corresponding to edge e_i at time-window W_t is, we use the following condition:

$$R_i^t = \begin{cases} A & \text{if } i + t \equiv 0 \pmod 2, \\ B & \text{else.} \end{cases}$$

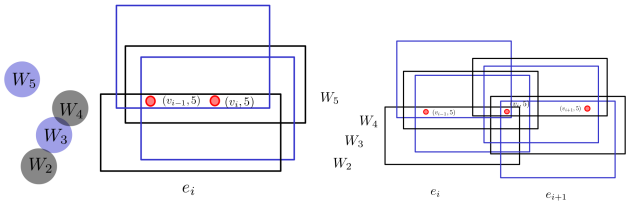
Now we define where the points are placed. We use the following conditions.

a. If an edge $e_i = v_{i-1}v_i$ appears at time t we add vertex appearances $(v_{i-1}, t), (v_i, t)$ to all of the rectangles $R_i^{t'}$, where $t - \Delta + 1 \leq t' \leq t$, if they exist. Equivalently, we add the vertex appearances $(v_{i-1}, t), (v_i, t)$ in the intersection of the rectangles corresponding to the edge e_i in the time windows $W_{t-\Delta+1}, \ldots, W_t$. For an example see Figure 5a.

b. If an edge e_i does not appear at time t, then the time vertices $(v_{i-1}, t), (v_i, t)$ are not included in the rectangles $R_i^{t'}$ $(t - \Delta + 1 \leq t' \leq t)$, if they exist.

c. If two adjacent edges e_i, e_{i+1} appear at the same time t, we add to the intersection of the rectangles $R_i^{t'}, R_{i+1}^{t'}$ the vertex (v_i, t), where $t - \Delta + 1 \leq t' \leq t$. For an example see Figure 5b.

It is straightforward to verify that finding the minimum hitting set of the range space is equivalent to finding the minimum Δ-TVC for $\mathcal{G}$. On the constructed range space we use the local search algorithm from (Mustafa and Ray 2010) which proves our result. □



(a) Edge e_i appearing at time 5 and the corresponding placement of vertex appearances to the rectangles, when $\Delta = 4$.

(b) Edges e_i and e_{i+1} appearing at time 5 and the corresponding vertex appearances placement in the rectangles, when $\Delta = 4$.

Figure 5: Example of placement of the vertices into the rectangles when $\Delta = 4$.

4 Algorithms for Bounded Degree Temporal Graphs

In this section we extend our focus from temporal graphs with underlying paths or cycles to instances of Δ-TVC with more general degree restrictions.

In particular we present an algorithm for solving Δ-TVC exactly in time that is single exponential in the number of edges of the underlying graph, then use this algorithm to give a $(d-1)$-factor approximation algorithm (where d is the maximum vertex degree in any time step) and finally give an FPT-algorithm parameterized by the size of a solution. For the approximation algorithm in particular the following generalized notion (sub)instances will be useful, already when formulating the exact exponential time algorithm.

Definition 7 (Partial Δ-TVC)**.** Let (G, λ) be a temporal graph. An instance of Partial Δ-TVC is given by $(G, \lambda, \ell, h, \alpha, \beta)$ where $\ell : E(G) \to [T]$ and $h : E(G) \to [T]$ map each edge to the starting time of its *lowest uncovered window* and *highest uncovered window* respectively, and $\alpha, \beta \in [T]$ are the *covering start* and *end* respectively. The task is to find a cardinality minimal temporal vertex subset $\mathcal{C}$ such that for every edge $e \in E(G)$ and every time window W_i with $\ell(e) \leq i \leq h(e)$ if $e \in E[W_i]$ then there is some $(v, t) \in \mathcal{C}[W_i]$ such that $v \in e, t \in \lambda(e)$, and additionally for all $(v, t) \in \mathcal{C}$, $\alpha \leq t \leq \beta$.

Obviously Partial Δ-TVC generalizes Δ-TVC by letting $\ell(e) = 0$ and $h(e) = T - \Delta + 1$ for all $e \in E(G)$, and $\alpha = 0$ and $\beta = T$.

4.1 Exact Algorithm

In the following we denote by d_G the degree of the underlying graph of the considered instances of Partial Δ-TVC. We can give a dynamic programming algorithm with running time $\mathcal{O}(T\Delta^{\mathcal{O}(|E(G)|)})$ as the next theorem states.

Theorem 8. *For every $\Delta \geq 2$, a solution to* Partial Δ-TVC *can be computed in time complexity $\mathcal{O}(Tc^{\mathcal{O}(|E(G)|)})$ where $c = \min\{2^{d_G}, \Delta\}$ and T is the life time of the temporal graph in the instance.*

In fact the same algorithm and analysis gives a running time bound which is single exponential in the *maximum Δ-window vertex degree* d_Δ which is the maximum vertex degree in any part of G that appears together in an arbitrary Δ-time window. Observe that $d_\Delta \leq d_G$ but there are also instances in which d_G is much larger than d_Δ:

Remark: The above algorithm solves Partial Δ-TVC *in* $\mathcal{O}(Tc^{\mathcal{O}(|E(G)|)})$ *where* $c = \min\{2^{d_\Delta}, \Delta\}$.

Note that this algorithm also has implications for the parameterized complexity (Cygan et al. 2015; Downey and Fellows 2013; Niedermeier 2006) of Δ-TVC.

Corollary 9. *For every $\Delta \geq 2$, Δ-TVC can be solved in time in $\mathcal{O}(Tc^{|E(G)|})$ where $c = \min\{2^{\mathcal{O}(|E(G)|)}, \Delta\}$, and thus Δ-TVC is in* FPT *parameterized by $|E(G)|$.*

4.2 Approximation Ratio Better Than d

Next we turn to our approximation algorithms; recall that a d-factor approximation is known (Akrida et al. 2020). The idea of this algorithm is simple; solve each subinstance induced by an edge independently and optimally and then combine these solutions. At each temporal vertex at most d time-edges which were considered separately in subinstances by the approximation algorithm occur. To cover each of these edges we might have chosen the 'wrong' endpoint in a subinstance rather than the shared endpoint.

Now our new exact algorithm for solving instances can be used to mitigate the error we can make at high degree vertices. For instance, if we build our subinstances by iteratively covering paths with two edges (P_3) instead of single edges we will incur an error of at most $d-1$ at vertices which are centers of at least one P_3 which was chosen as a subinstance.

Based on this idea we can formulate a $(d-1)$-approximation.

Description of the Algorithm We iteratively extend an initially empty set $\mathcal{X}$ to a sliding Δ-window TVC in the following way: While there is some $e \in E(G)$ with an occurrence that is not covered in some time window in the lifetime of $\mathcal{G}$ we have to extend $\mathcal{X}$; otherwise $\mathcal{X}$ is a Δ-TVC which we can return. We proceed in two phases:

Phase 1: While we find two such edges e_1, e_2 that are adjacent and appear at the same time step and both appearances are not covered in some time window W_i then we consider the following subinstances of Partial Δ-TVC: Let S be the set of all time steps in which e_1 and e_2 appear together and are both not covered by $\mathcal{X}$ in some time window W_i with $i \in [\min S - \Delta + 1, \max S - \Delta + 1]$. We can subdivide S into subsets $S_1, \ldots, S_k$ with $k \leq T$ such that S_1 contains the smallest elements of S such that there is no gap of at least $2\Delta - 1$ between its elements, S_2 contains the smallest elements of S between the first and second gap of at least $2\Delta - 1$, and so on. Now we consider the subinstances given by $(G[e_1, e_2], \lambda', \ell, h, \min S_i, \max S_i)$ with λ' defined as the restriction of λ to $\{e_1, e_2\} \cap \lambda^{-1}([\min S_i - \Delta + 1, \max S_i - \Delta + 1])$, $\ell(e_1) = \ell(e_2)$ is the smallest time step t such that $(e_1, \min S_i)$ and $(e_2, \min S_i)$ are not covered in time window W_t by $\mathcal{X}$, and $h(e_1) = h(e_2)$ is the largest time step t such that $(e_1, \max S_i)$ and $(e_2, \max S_i)$ are not covered in time window W_t by $\mathcal{X}$. We use the algorithm from Section 4.1 to solve these subinstances and extend $\mathcal{X}$ by the union of the solutions.

Phase 2: If no such edges are adjacent and appear at the same time step and both appearances are not covered in some time window W_i let F be the set of edges $e \in E(G)$ with occurrences which is not covered in some time window in the lifetime of $\mathcal{G}$ by $\mathcal{X}$. We consider the following subinstances of Partial Δ-TVC: For $e \in F$, let S^e be the set of all time steps in which e appears and is not covered by $\mathcal{X}$ in some time window W_i with $i \in [\min S^e - \Delta + 1, \max S^e - \Delta + 1]$ and $t \in S^e$. We can subdivide S^e into subsets $S^e_1, \ldots S^e_k$ with $k \leq T$ such that S^e_1 contains the smallest elements of S^e such that there is no gap of at least $2\Delta - 1$ between its elements, S^e_2 contains the smallest elements of S^e between the first and second gap of at least $2\Delta - 1$, and so on. Now we consider the subinstances given by $(G[e], \lambda', \ell, h, \min S^e_i, \max S^e_i)$ with λ' defined as the restriction of λ to $\{e\} \cap \lambda^{-1}([\min S^e_i - \Delta + 1, \max S^e_i + \Delta])$, $\ell(e)$ is the smallest time step t such that $(e, \min S^e_i)$ is not covered in time window W_t by $\mathcal{X}$, and $h(e)$ is the largest time step t such that $(e, \max S^e_i)$ is not covered in time window W_t by $\mathcal{X}$. We use the algorithm from Section 4.1 to solve these subinstances and extend $\mathcal{X}$ by the union of the solutions.

It follows from the above construction that the produced set $\mathcal{X}$ of vertex appearances is a Δ-TVC of $\mathcal{G}$. Furthermore, using a double counting argument, we can show that the approximation ratio is at most $d-1$.

Running Time The number of subinstances considered in Phase 1 is easily bounded by the number of combinations of two edges in $E(G)$ multiplied by T. Similarly the number of subinstances considered in Phase 2 is bounded by the number of edges in $E(G)$ multiplied by T. The subinstances require a running time of $\mathcal{O}(T)$ to solve. Thus the overall running time lies in $\mathcal{O}(|E(G)|^2T^2)$.

Overall this shows the desired approximation result.

Theorem 10. *For every $\Delta \geq 2$ and $d \geq 3$, Δ-TVC can be $(d-1)$-approximated in time $\mathcal{O}(|E(G)|^2T^2)$.*

4.3 An FPT algorithm with respect to the solution size

Our final result settles the complexity of Δ-TVC from the viewpoint of parameterized complexity theory (Cygan et al. 2015; Downey and Fellows 2013; Niedermeier 2006) with respect to the standard parameterization of the size of an optimum solution.

Theorem 11. *For every $\Delta \geq 2$, Δ-TVC can be solved in $\mathcal{O}((2\Delta)^k Tn^2)$ time, where k is the size of an optimum solution. In particular Δ-TVC is in* FPT *parameterized by k.*

References

Aaron, E.; Krizanc, D.; and Meyerson, E. 2014. DMVP: Foremost waypoint coverage of time-varying graphs. In *Proceedings of the 40th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, 29–41.

Akrida, E. C.; Gasieniec, L.; Mertzios, G. B.; and Spirakis, P. G. 2016. Ephemeral networks with random availability of links: The case of fast networks. *Journal of Parallel and Distributed Computing*, 87: 109–120.

Akrida, E. C.; Gasieniec, L.; Mertzios, G. B.; and Spirakis, P. G. 2017. The complexity of optimal design of temporally connected graphs. *Theory of Computing Systems*, 61(3): 907–944.

Akrida, E. C.; Mertzios, G. B.; Spirakis, P. G.; and Zamaraev, V. 2020. Temporal vertex cover with a sliding time window. *Journal of Computer and System Sciences*, 107: 108–123.

Bentert, M.; Himmel, A.-S.; Molter, H.; Morik, M.; Niedermeier, R.; and Saitenmacher, R. 2018. Listing all maximal k-plexes in temporal graphs. In *Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 41–46.

Bui-Xuan, B.-M.; Ferreira, A.; and Jarry, A. 2003. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2): 267–285.

Casteigts, A.; and Flocchini, P. 2013a. Deterministic algorithms in dynamic networks: Formal models and metrics. Technical report, Defence R&D Canada.

Casteigts, A.; and Flocchini, P. 2013b. Deterministic algorithms in dynamic networks: Problems, analysis, and algorithmic tools. Technical report, Defence R&D Canada.

Casteigts, A.; Flocchini, P.; Quattrociocchi, W.; and Santoro, N. 2012. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5): 387–408.

Casteigts, A.; Himmel, A.; Molter, H.; and Zschoche, P. 2021. Finding temporal paths under waiting time constraints. *Algorithmica*, 83(9): 2754–2802.

Chen, J.; Molter, H.; Sorge, M.; and Suchý, O. 2018. Cluster editing in multi-layer and temporal graphs. In Hsu, W.; Lee, D.; and Liao, C., eds., *Proceedings of the 29th International Symposium on Algorithms and Computation (ISAAC)*, volume 123, 24:1–24:13.

Clementi, A. E. F.; Macci, C.; Monti, A.; Pasquale, F.; and Silvestri, R. 2010. Flooding time of edge-markovian evolving graphs. *SIAM Journal on Discrete Mathematics*, 24(4): 1694–1712.

Cygan, M.; Fomin, F. V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer. ISBN 978-3-319-21274-6.

de Berg, M.; and Khosravi, A. 2010. Optimal binary space partitions in the plane. In *Computing and combinatorics*, volume 6196 of *Lecture Notes in Computer Science*, 216–225. Springer, Berlin.

Downey, R. G.; and Fellows, M. R. 2013. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. London: Springer. ISBN 978-1-4471-5558-4.

Enright, J. A.; Meeks, K.; Mertzios, G. B.; and Zamaraev, V. 2021. Deleting edges to restrict the size of an epidemic in temporal networks. *Journal of Computer and System Sciences*, 119: 60–77.

Erlebach, T.; Hoffmann, M.; and Kammer, F. 2021. On temporal graph exploration. *Journal of Computer and System Sciences*, 119: 1–18.

Ferreira, A. 2004. Building a reference combinatorial model for MANETs. *IEEE Network*, 18(5): 24–29.

Flocchini, P.; Mans, B.; and Santoro, N. 2009. Exploration of periodically varying graphs. In *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC)*, 534–543.

Ghosal, S.; and Ghosh, S. C. 2015. Channel assignment in mobile networks based on geometric prediction and random coloring. In *Proceedings of the 40th IEEE Conference on Local Computer Networks (LCN)*, 237–240.

Giakkoupis, G.; Sauerwald, T.; and Stauffer, A. 2014. Randomized rumor spreading in dynamic graphs. In *Proceedings of the 41st International Colloquium on Automata, Languages and Programming (ICALP)*, 495–507.

Heeger, K.; Hermelin, D.; Mertzios, G. B.; Molter, H.; Niedermeier, R.; and Shabtay, D. 2021. Equitable scheduling on a single machine. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, 11818–11825. AAAI Press.

Himmel, A.; Molter, H.; Niedermeier, R.; and Sorge, M. 2017. Adapting the Bron-Kerbosch algorithm for enumerating maximal cliques in temporal graphs. *Social Network Analysis and Mining*, 7(1): 35:1–35:16.

Holme, P.; and Saramäki, J., eds. 2013. *Temporal networks*. Springer.

Ileri, C. U.; Ural, C. A.; Dagdeviren, O.; and Kavalci, V. 2016. On vertex cover problems in distributed systems. In *Advanced Methods for Complex Network Analysis*, 1–29. IGI Global.

Kavalci, V.; Ural, A.; and Dagdeviren, O. 2014. Distributed vertex cover algorithms for wireless sensor networks. *International Journal of Computer Networks & Communications*, 6(1): 95–110.

Kempe, D.; Kleinberg, J.; and Kumar, A. 2002. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4): 820–842.

Klobas, N.; Mertzios, G. B.; Molter, H.; Niedermeier, R.; and Zschoche, P. 2021. Interference-free walks in time: Temporally disjoint paths. In Zhou, Z., ed., *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, 4090–4096.

Leskovec, J.; Kleinberg, J. M.; and Faloutsos, C. 2007. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1).

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

Mertzios, G. B.; Molter, H.; Renken, M.; Spirakis, P. G.; and Zschoche, P. 2021. The complexity of transitively orienting temporal graphs. In *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 75:1–75:18.

Mertzios, G. B.; Molter, H.; and Zamaraev, V. 2021. Sliding window temporal graph coloring. *Journal of Computer and System Sciences*, 120: 97–115.

Michail, O.; and Spirakis, P. G. 2016. Traveling salesman problems in temporal graphs. *Theoretical Computer Science*, 634: 1–23.

Michail, O.; and Spirakis, P. G. 2018. Elements of the theory of dynamic networks. *Communications of the ACM*, 61(2): 72–72.

Mustafa, N. H.; and Ray, S. 2010. Improved results on geometric hitting set problems. *Discrete and Computational Geometry*, 44(4): 883–895.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and its Applications. Oxford: Oxford University Press. ISBN 978-0-19-856607-6.

Richter, S.; Helmert, M.; and Gretton, C. 2007. A Stochastic Local Search Approach to Vertex Cover. In *Proceedings of the 30th Annual German Conference on Artificial Intelligence (KI)*, 412–426.

Tang, J. K.; Musolesi, M.; Mascolo, C.; and Latora, V. 2010. Characterising temporal distance and reachability in mobile and online social networks. *Computer Communication Review*, 40(1): 118–124.

Viard, T.; Latapy, M.; and Magnien, C. 2016. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609: 245–252.

Yu, F.; Bar-Noy, A.; Basu, P.; and Ramanathan, R. 2013. Algorithms for channel assignment in mobile wireless networks using temporal coloring. In *Proceedings of the 16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems (MSWiM)*, 49–58.

Zschoche, P.; Fluschnik, T.; Molter, H.; and Niedermeier, R. 2020. The complexity of finding small separators in temporal graphs. *Journal of Computer and System Sciences*, 107: 72–92.