Algorithmica (2023) 85:805–830
https://doi.org/10.1007/s00453-022-01021-y

Eulerian Walks in Temporal Graphs

Andrea Marino[1] · **Ana Silva**[1,2]

Received: 15 October 2021 / Accepted: 26 July 2022 / Published online: 25 August 2022

Abstract
An Eulerian walk (or Eulerian trail) is a walk (resp. trail) that visits every edge of a graph G at least (resp. exactly) once. This notion was first discussed by Leonhard Euler while solving the famous Seven Bridges of Königsberg problem in 1736. But what if Euler had to take a bus? In a temporal graph (G, λ), with $\lambda : E(G) \to 2^{[\tau]}$, an edge $e \in E(G)$ is available only at the times specified by $\lambda(e) \subseteq [\tau]$, in the same way the connections of the public transportation network of a city or of sightseeing tours are available only at scheduled times. In this paper, we deal with temporal walks, local trails, and trails, respectively referring to edge traversal with no constraints, constrained to not repeating the same edge in a single timestamp, and constrained to never repeating the same edge throughout the entire traversal. We show that, if the edges are always available, then deciding whether (G, λ) has a temporal walk or trail is polynomial, while deciding whether it has a local trail is NP-complete even if $\tau = 2$. In contrast, in the general case, solving any of these problems is NP-complete, even under very strict hypotheses. We finally give XP algorithms parametrized by τ for walks, and by $\tau + tw(G)$ for trails and local trails, where $tw(G)$ refers to the treewidth of G.

Keywords Temporal graphs · Eulerian walks · Eulerian trails · Edge cover · Bounded treewidth · Parameterized complexity

Andrea Marino and Ana Silva have contributed equally to this work.

✉ Andrea Marino
andrea.marino@unifi.it

Ana Silva
anasilva@mat.ufc.br

[1] Dipartimento di Statistica, Informatica, Applicazioni, Università degli Studi di Firenze, Viale Morgagni 65, 50134 Firenze, Italy

[2] Departamento de Matemática, Universidade Federal do Ceará, Avenida Mister Hull s/n Campus do Pici, Fortaleza, Ceará 10587, Brazil

1 Introduction

An Eulerian walk (or Eulerian trail) is a walk (resp. trail) that visits every edge of a graph G at least once (resp. exactly once). The Eulerian trail notion was first discussed by Leonhard Euler while solving the famous Seven Bridges of Königsberg problem in 1736, where one wanted to pass by all the bridges over the river Preger without going twice over the same bridge. Imagine now a similar problem, where you have a set of tourist sights linked by possible routes. If the routes themselves are also of interest, a sightseeing tourism company might want to plan visits on different days that cover all the routes. One could do that with no constraints at all (thus performing a walk), or with the very strict constraint of never repeating a route (thus getting a trail), or constraining oneself to at least not repeating the same route on the same day (thus getting what we called a local trail). If we further assume that some routes might not be always accessible, we then get distinct problems defined on temporal graphs.

In a temporal graph (G, λ) with lifetime τ, we have $\lambda : E(G) \to 2^{[\tau]}$, and an edge $e \in E(G)$ is available only at the times specified by $\lambda(e) \subseteq [\tau]$, in the same way the connections of the public transportation network of a city or of sightseeing tours are available only at scheduled times. In this scenario, paths and walks are valid only if they traverse a sequence of adjacent edges $e_1, \ldots, e_k$ at non-decreasing times $t_1 \leq \ldots \leq t_k$, respectively, with $t_i \in \lambda(e_i)$ for every $i \in [k]$ (similarly, one can consider strictly increasing sequences, i.e. with $t_1 < \ldots < t_k$). The research on temporal graphs has attracted a lot of attention in the past decade (we refer the reader to the surveys of [21, 24], and the seminal paper of [19]). They have appeared also under different names, e.g. as time-varying graphs [6], as evolving networks [4]), and as link streams [21].

Several translations of Eulerian trails and walks are possible in temporal terms, depending on the constraints we consider. In particular, we study the following variations. Below, all the walks and trails are implicitly considered to be temporal, as defined in the previous paragraph.

Problem Given a temporal graph (G, λ), we consider the following problems:

- EULERIAN WALK: deciding whether (G, λ) has an Eulerian walk, i.e. a walk traversing each edge of G at least once.
- EULERIAN LOCAL TRAIL: deciding whether (G, λ) has an Eulerian local trail, i.e. a walk traversing each edge of G at least once, and at most once in each timestamp.
- EULERIAN TRAIL: deciding whether (G, λ) has an Eulerian trail, i.e. a walk traversing each edge of G exactly once.

We also consider the related problems where the walks/trails are closed (first vertex equal to the last one), respectively referring to them as EULERIAN CLOSED WALK, EULERIAN LOCAL TOUR, and EULERIAN TOUR. Finally, for all of the above problems, we add the prefix STRICT to refer to the variation in which walks must be strictly increasing sequences of edges. Observe that, when $\tau = 1$, then both EULERIAN TRAIL and EULERIAN LOCAL TRAIL degenerate into the original formulation of the Seven Bridges of Königsberg problem. This is why we think they appear to be more natural adaptations of the static version of the problem.

One of the concepts closest to ours is the one defined by [26], where he gives a polynomial-time algorithm to check the existence of an Eulerian closed walk (i.e. a *tour*) in dynamic graphs. However, the dynamic graph model is quite different from the temporal graph model used in this paper, as pointed out by [24]. Indeed, looking at the corresponding time-expanded graph related to the one of [26], temporal edges can go back in time and the graph is infinite. Nevertheless, the results presented there seemed to point towards the polynomiality of the problems investigated here, as observed by [24]: "the results proved for it [the dynamic graph model] are resounding and possibly give some first indications of what to expect when adding to combinatorial optimization problems a time dimension". We found however that this is not the case, as we will show that even EULERIAN WALK turns out to be NP-complete on temporal graphs. Taking inspiration by [26], we also define a dynamic-based temporal graph as a temporal graph whose edges are always available, and we analyze the complexity of the above problems on these particular instances.

In this paper we prove the following results. These are summarized in Table 1, which also reports some recent results that will be discussed shortly.

Theorem 1 *Given a temporal graph* (G, λ) *with lifetime* τ*,*

1. EULERIAN WALK *is NP-complete, even if either each snapshot of* (G, λ) *is a forest with a constant number of edges, or each edge appears at most 3 times. Also, it is polynomial-time solvable if* (G, λ) *is dynamic-based, and is in XP when parameterized by* τ*.*
2. EULERIAN LOCAL TRAIL *is NP-complete for each* $\tau \geq 2$*, even if* (G, λ) *is dynamic-based. It is in XP when parameterized by* $\tau + tw(G)$*.*
3. EULERIAN TRAIL *is NP-complete for each* $\tau \geq 2$*. It is polynomial if* (G, λ) *is dynamic-based. It is in XP when parameterized by* $\tau + tw(G)$*.*

The same results hold for tours, i.e. for EULERIAN CLOSED WALK*,* EULERIAN LOCAL TOUR*, and* EULERIAN TOUR*.*

Theorem 1 gives a complete taxonomy of our problems, also focusing on the possibility of getting polynomial-time algorithms when we have a small lifetime τ. In particular, for EULERIAN TRAIL and EULERIAN LOCAL TRAIL, since they become polynomial when $\tau = 1$, the bound for τ is optimal, giving us a complete dichotomy with respect to the lifetime of (G, λ). In contrast, EULERIAN WALK is easily solvable for every fixed τ, meaning that walks are easier than trails even on the temporal context. We also show that EULERIAN TRAIL and EULERIAN LOCAL TRAIL are solvable in XP time if parameterized by $\tau + tw(G)$. Our negative results for constant τ combined with known negative results for constant $tw(G)$ (see Table 1) exclude the possibility of XP algorithms if parameterized just by τ or $tw(G)$. An FPT algorithm parameterized by $\tau + tw(G)$ could still exist.

It is important to remark that none of the variations we considered immediately implies any of the others. We will show indeed that the property of being Eulerian for the static base graph G is in general a necessary but not sufficient condition for the existence of an Eulerian trail, becoming sufficient only if we restrict to dynamic-based temporal graphs. In the case of Eulerian local trail, we will see that this property is not even necessary. In addition, if only strictly increasing temporal walks/trails are

Table 1 Our results concerning Problem 1

	WALK	LOCAL TRAIL	TRAIL
GENERAL	NP-c $\forall$ fixed $k \geq 3$, XP by τ	NP-c $\forall$ fixed $\tau \geq 2$ XP by $\tau + tw(G)$	NP-c $\forall$ fixed $\tau \geq 2$, XP by $\tau + tw(G)$ FPT by $k + imw(G, \lambda)$
DYNAMIC- BASED	Poly†	NP-c $\forall$ fixed $\tau \geq 2$	Poly*
TREEWIDTH $\leq w$	NP-c $\forall$ fixed $k \geq 4$, even if $w = 1$	NP-c $\forall$ fixed $k \geq 4$, even if $w = 1$	Poly (trivial) when $w = 1$ NP-c $\forall$ fixed $k \geq 4$, even if $w = 2$

For general temporal graphs (first row), for dynamic-based temporal graphs (second row), for temporal graphs whose base graph has bounded treewidth (last row). k refers to the maximum number of appearances of an edge. † corresponds to deciding whether G is connected. Blue are direct applications or easy implications of results by [5].

* Corresponds to deciding whether G has an Eulerian trail. All the results to STRICT EULERIAN WALK, STRICT EULERIAN LOCAL TRAIL, STRICT EULERIAN TRAIL, except that we $\tau = \Omega(m)$ is necessary, with $m = |E(G)|$, as when $\tau < m$ the answer is trivially NO

allowed, then our reductions for the first row of Table 1 can be easily modified, thus giving NP-completeness results also in this case. Observe that in this case a necessary condition for a positive answer is $\tau \geq |E(G)|$; this is why we do not have the same bounds for τ as in Theorem 1.

Corollary 2 *(i)* STRICT EULERIAN WALK, *(ii)* STRICT EULERIAN LOCAL TRAIL, *(iii)* STRICT EULERIAN TRAIL *are NP-complete in a general temporal graph* (G, λ) *with lifetime* $\tau = \Omega(|E(G)|)$.

Also in the case of dynamic-based temporal graphs (second row of Table 1), the polynomiality is preserved for strictly increasing Eulerian walks and Eulerian trails and we leave open the question whether STRICT EULERIAN LOCAL TRAIL is still NP-complete on dynamic-based temporal graphs.

As a byproduct of our reductions we get the following result about static graphs, which can be of independent interest.

Corollary 3 *Given a graph G, deciding whether the edges of G can be covered with two trails is NP-complete.*

A preliminary version of this paper appeared in [23]. With respect to that version, we now present complete proofs of Theorems 5 and 8, we add an XP algorithm parameterized by treewidth and lifetime (Sect. 6), and we discuss the dynamic graph result by [26] in Appendix A.

Related work

EULERIAN WALK is related to the TEXP problem investigated by [25], which consists of, given a temporal graph (G, λ), finding a strictly increasing temporal walk that visits all vertices in G (possibly, more than once) whose arrival time is minimum. [25] prove that TEXP is NP-complete and even not approximable unless ¶ = NP. This is in stark contrast with the static version of the problem, which can be trivially solved in linear time. A lot of research has been devoted to temporal exploration, e.g. bounding the arrival time of such walks in special instances [11, 13] and extending previous results in the case of non-strictly increasing paths [12]. Akrida et al. [1] proved that TEXP is NP-complete even when restricted to temporal stars in which each edge appears at most k times, for all fixed $k \geq 6$. This result has been improved by [5] to all fixed $k \geq 4$. On the other hand, Akrida et al. [1] showed that if each edge appears at most $k = 3$ times, then the problem is polynomial-time solvable on temporal stars. Observe that, in a star, passing by all the leaves translates into passing by all the edges. Therefore their result implies NP-completeness for STRICT EULERIAN WALK with the same constraints as before. Note that in contrast EULERIAN WALK is trivial on stars since one can always go back to the central vertex. Nevertheless, as will be discussed in Sect. 6, EULERIAN WALK is NP-complete on general trees by a modification of the reduction presented by [5], as highlighted by Table 1. Notice also that, unlike stars, our main theorem says that EULERIAN WALK on general temporal graphs is NP-complete when edges appear at most 3 times.

Now, note that STRICT EULERIAN WALK is equivalent to (STRICT) EULERIAN LOCAL TRAIL. Hence, still from the previous paragraph, we get that (STRICT) EULERIAN LOCAL TRAIL is NP-complete on stars. In particular, we get that EULERIAN LOCAL TRAIL is NP-complete when restricted to temporal graphs where each edge appears at most k times, for all fixed $k \geq 4$. We improve this bound for $k = 2$ even when the temporal graph is dynamic based, and present an algorithm that runs in XP time when parametrized by $\tau + tw(G)$.

Concerning EULERIAN TRAIL, to the best of our knowledge, there is only one other paper investigating this problem, that appeared in the same volume as the preliminary version of this paper and focuses on the EULERIAN TRAIL variation, giving independent and broadly different results [5]. In particular, other than proving the results on TEXP mentioned above, they prove that EULERIAN TRAIL is NP-complete even if each edge appears at most k times, for every fixed $k \geq 3$. Observe that our result improves that to $k = 2$, since we prove it is NP-complete even if the lifetime is 2. Nevertheless, even though their reduction produces a temporal graph with unbounded lifetime, it also gives a base graph with very simple structure (a set of triangles intersecting in a single vertex, which is a graph with treewidth 2). In addition, they also introduce a parameter for temporal graphs, that they called *interval-membership width* (denoted by $imw(G, \lambda)$), and provide an algorithm FPT when parametrized by $k + imw(G, \lambda)$, as also reported in Table 1. As $k + imw(G, \lambda)$ is not related to $\tau + tw(G)$, our algorithm XP in $\tau + tw(G)$ is of independent interest.

When considering dynamic-based temporal graphs, as edges are assumed to be always available during the lifetime τ, we could relate our problems to several other problems on static graphs. A closely related one would be the Chinese Postman problem, where the edges of the graph have positive weights and one wants to find an

Eulerian closed walk on G with minimum weight; in other words, one wants to add copies of existing edges in order to obtain an Eulerian graph of minimum sum weight. Even if we regard the Chinese Postman problem where the weights are all equal to 1, this is very different from our approach since for us, repetition of a long common trail in different snapshots does not make the solution worse, while it would when considering the Chinese Postman problem. It is easy to see though that the solution for the Chinese Postman problem would give us an upper bound for the amount of time spent on an Eulerian local tour of a dynamic-based graph, as we could start a new trail on a new snapshot whenever an edge repetition was detected. The Chinese Postman problem is largely known to be polynomial [18], and some variations that take time into consideration have been investigated, mostly from the practical point of view (see e.g. [7, 27, 28]), but none of which is equivalent to our problems. Among these, we mention the so-called HCPP (Hierarchical Chinese Postman Problem), where the set E of all the edges is partitioned into clusters, i.e. $\{E_1, \ldots, E_h\}$ such that $\cup_{i=1}^{h} E_i = E$ and for $i \neq j$ $E_i \cap E_j = \emptyset$, and there is a partial order $<$ between clusters. The HCPP aims to find a path of minimum weight passing through all the edges and such that, if an edge in E_j is traversed, all the edges in E_i for every $i < j$ must be previously traversed. This has been shown to be NP-hard in general and polynomial if the order is linear and other conditions are met [10]. As it can be seen, this problem is very different from our problems for several reasons: a walk is free to traverse an edge E_j and then an edge in E_i with $i < j$, each edge belongs to just one time stamp, and each walk must traverse *all* the edges in the previous timestamps.

The problem of trying to obtain an Eulerian subgraph (as opposed to a supergraph, as was the case in the previous paragraph) has also been studied. [8] study a family of problems where the goal is to make a static graph Eulerian by a minimum number of deletions. They completely classify the parameterized complexity of various versions of the problem: vertex or edge deletions, undirected or directed graphs, with or without the requirement of connectivity. Also, [15] study the parameterized complexity of the following Euler subgraph problems: (i) Largest Euler Subgraph - for a given graph G and integer parameter k, does G contain an induced Eulerian subgraph with at least k vertices?; and (ii) Longest Circuit - for a given graph G and integer parameter k, does G contain an Eulerian subgraph with at least k edges?

Eulerian Local Trail on dynamic-based graphs is actually more closely related to the problem of covering the edges of a graph with the minimum number of (not necessarily disjoint) trails, whereas the aforementioned problems are more concerned with either minimizing edge repetitions or maximizing the subgraph covered by a single trail. Even if the trail cover problem can be so naturally defined and involve such a basic structure as trail, up to our knowledge it has not been previously investigated yet. Note that Eulerian Local Trail is slightly different from trail cover, since we also require that together the trails form a walk. In any case, a small modification of our proof of Theorem 1.1 implies that deciding whether the edges of a graph can be covered with at most two trails is NP-complete (Corollary 3). We mention that the vertex version of this problem, namely the path cover problem, has been largely investigated (see e.g. [2, 17, 22]).

2 Preliminaries

We use standard definitions and notation of graph theory; we refer the unfamiliar reader to [29]. A graph G has an Eulerian tour (trail) if and only if G has at most one non-trivial component and all the vertices have even degree (at most two vertices have odd degree) (see [14]). A graph is called *Eulerian* if it has an Eulerian tour.

Concerning temporal graphs, we use and extend the notation in [24]. A temporal graph is a graph together with a function on the edges saying when each edge is active; more formally, a *temporal graph* is a pair (G, λ), where $\lambda : E(G) \to 2^{\mathbb{N}-\{0\}}$. Here, we consider only finite temporal graphs, i.e., graphs such that $\max \bigcup_{e \in E(G)} \lambda(e)$ is defined. This value is called the *lifetime of* (G, λ) and denoted by τ. Given $i \in [\tau]$, we define the *snapshot* G_i as being the subgraph of G containing exactly the edges active in time i; more formally, $V(G_i) = V(G)$ and $E(G_i) = \{e \in E(G) \mid i \in \lambda(e)\}$. An element $(e, t) \in E(G) \times [\tau]$ is called a *temporal edge*. In what follows, in order to make the reading more fluid, we often talk about "occurrence of an edge". When this is done we are implicitly referring to the corresponding temporal edge; for instance, by "edge e occurs in G_i" we mean that $i \in \lambda(e)$ (and hence (e, i) is a temporal edge).

Given vertices v_0, v_k in a graph G, a v_0, v_k-*walk* in G is an alternating sequence $(v_0, e_1, v_1, \ldots, e_k, v_k)$ of vertices and edges such that e_i goes from v_{i-1} to v_i for $i \in \{1, \ldots, k\}$. We define a walk in a temporal graph similarly, except that a walk cannot go back in time. More formally, given a temporal graph (G, λ), a sequence $W = (v_0, (e_1, t_1), v_1, \ldots, (e_k, t_k), v_k)$ of vertices and temporal edges is called a *temporal* v_0, v_k-*walk* if $(v_0, e_1, v_1, \ldots, e_k, v_k)$ is a walk in G, $t_i \in \lambda(e_i)$ for every $i \in [k]$, and $t_1 \leq \ldots \leq t_k$. It is *closed* if it starts and finishes on the same vertex of G, i.e., if $v_0 = v_k$, and it is *strict* if $t_1 < \ldots < t_k$.

We say that a temporal walk W is a *trail* if there are no two occurrences of the same edge of G in W. We say that W is a *local trail* if there are no two occurrences of the same edge of G in the same snapshot, i.e., if W restricted to G_i is a trail in G for every $i \in [\tau]$. A closed (local) trail is also called a *(local) tour*. Finally, a temporal walk W is called *Eulerian* if at least one occurrence of each edge of G appears at least once in W. Observe that, by definition, an Eulerian temporal trail visits every edge of G exactly once. From now on, we omit the word "temporal" as all treated walks/trails are temporal.

A *dynamic-based graph* is a temporal graph (G, λ) where the edges are always available, i.e. $\lambda(e) = [\tau]$ for each $e \in E(G)$.[1] We denote a dynamic-based graph simply by $(G, [\tau])$ where τ is the lifetime of the temporal graph.

3 Eulerian Walk

In this section we focus on Eulerian Walk, i.e. deciding if there is a temporal walk passing by each edge at least once, proving the results in Item 1 in Theorem 1,

[1] This is the reason why we use the term dynamic-based, as they are similar to the dynamic networks used in [26] when studying Eulerian trails, except that edges cannot go back in time and the lifetime is finite.

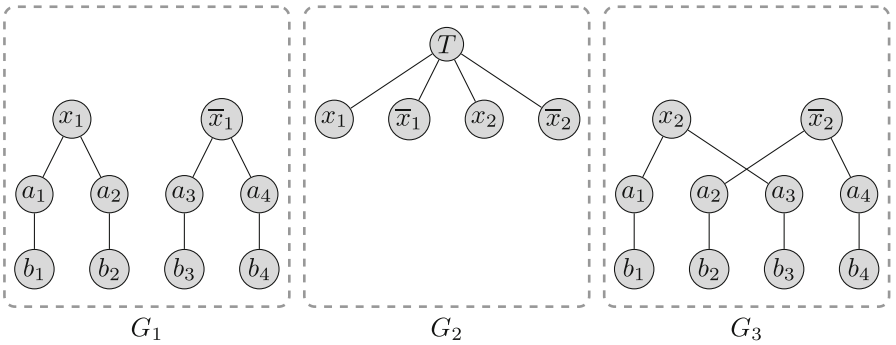


Fig. 1 First three snapshots of the construction. For simplicity, we represent only the non-trivial components of each snapshot. In this example, we have c_1 containing $(x_1 \vee x_2)$, c_2 containing $(x_1 \vee \overline{x}_2)$, c_3 containing $(\overline{x}_1 \vee x_2)$, and c_4 containing $(\overline{x}_1 \vee \overline{x}_2)$

summarized in the first column of Table 1. We start by presenting a simple algorithm that solves the problem in XP time when parameterized by the lifetime.

Lemma 4 *Given a temporal graph (G, λ) with lifetime τ, solving* Eulerian Walk *on (G, λ) can be done in time $O(\tau \cdot (n+m) \cdot n^{\tau-1})$, where $n = |V(G)|$ and $m = |E(G)|$.*

Proof Let $G_1, \ldots, G_\tau$ be the snapshots of G; note first that if $E(G_i)$ is empty, then this snapshot can be suppressed. Our problem reduces to deciding whether there is a choice of connected components $H_1, \ldots, H_\tau$ of $G_1, \ldots, G_\tau$, respectively, that together cover all the edges of G and is such that H_i intersects H_{i+1}, for each $i \in [\tau - 1]$. More formally, $H_1, \ldots, H_\tau$ must be such that: $H_i \subseteq V(G_i)$ and $G[H_i]$ is connected, for each $i \in [\tau]$; $V(G[H_i]) \cap V(G[H_{i+1}]) \neq \emptyset$, for each $i \in [\tau - 1]$; and for each edge $e \in E(G)$, there is at least one $j \in [\tau]$ such that $e \in E(G[H_i])$. As for each $i \in [\tau]$, there are at most n nodes in the intersections, there are at most $O(n^{\tau-1})$ choices. For each choice the test can be done in $O(\tau(n+m))$, obtaining $O(\tau(n+m)n^{\tau-1})$ running time.

In the following, we show that when τ is unbounded, deciding whether (G, λ) admits an Eulerian walk is NP-complete by reducing from 3-SAT. This is best possible because of the above lemma.

Theorem 5 *Given a temporal graph (G, λ), deciding whether (G, λ) admits an Eulerian walk is NP-complete, even if either each snapshot of (G, λ) is a forest with a constant number of edges, or each edge appears in at most 3 snapshots.*

Proof We make a reduction from 3-SAT. Let ϕ be a 3-CNF formula on variables $\{x_1, \ldots, x_n\}$ and clauses $\{c_1, \ldots, c_m\}$, and construct G as follows. For each clause c_i, add vertices $\{a_i, b_i\}$ to G and edge $a_i b_i$. Now consider a variable x_i, and let $c_{i_1}, \ldots, c_{i_p}$ be the clauses containing x_i positively, and $c_{j_1}, \ldots, c_{j_q}$ be the clauses containing x_i negatively. Add two new vertices $x_i, \overline{x}_i$ to G, and edges $\{x_i a_{i_k} \mid k \in [p]\} \cup \{\overline{x}_i a_{j_k} \mid k \in [q]\}$; denote the spanning subgraph of G formed by these edges by H_i, and let H_i' be

equal to H_i together with edges $\{a_i b_i \mid i \in \{i_1, \ldots, i_p, j_1, \ldots, j_q\}\}$. We can suppose that $\{i_1, \ldots, i_p\} \cap \{j_1, \ldots, j_q\} = \emptyset$ as otherwise the clauses in the intersection would always be trivially valid; thus we get that H_i, H'_i are forests. Finally, add a new vertex T and make it adjacent to every vertex in $\{x_i, \overline{x}_i \mid i \in [n]\}$.

We now describe the snapshots of (G, λ). See Fig. 1 to follow the construction. We first build 2 consecutive snapshots in (G, λ) related to x_i, for each $i \in [n]$. The first one is equal to H'_i, and the second one contains exactly the edges $\{Tx_i, T\overline{x}_i, Tx_{i+1}, T\overline{x}_{i+1}\}$ if $i < n$, and if $i = n$, then the second snapshot is equal to $G - \{a_j b_j \mid j \in [m]\}$; this can be done because this subgraph is connected. Denote by S^1_i, S^2_i the first and second snapshot of x_i, for each $i \in [n]$. Put these snapshots consecutively in timestamps 1 through $2n$, in the order of the indexing of the variables. For now, observe that only the last snapshot might not be a forest; this will be fixed later. We now prove that ϕ is a satisfiable formula if and only if (G, λ) admits an Eulerian walk.

Observe first that, since we are dealing with a walk, we are allowed to repeat edges as many times as we want; hence, if we visit any vertex inside a component of a snapshot, we can also visit the entire component. At the same time, it is not possible to visit more than one component within a snapshot. In the following we show the proof of equivalence.

First consider a satisfying assignment of ϕ. Now, construct an Eulerian walk as follows. Start by visiting all the edges in the component of H'_1 containing x_1, if x_1 is true, or the one containing $\overline{x}_1$, otherwise. Then, in S^2_1, jump to x_2 if x_2 is true, or to $\overline{x}_2$, otherwise. Repeat the process until reaching S^1_n, and at the last snapshot, visit all the edges in $G - \{a_j b_j \mid j \in [m]\}$. Because each clause c_i contains at least one true variable, we know that edge $a_i b_i$ is visited.

Now, consider an Eulerian walk W of (G, λ), and denote by W^j_i the walk W restricted to S^j_i, for each $i \in [n]$ and $j \in [2]$. We set x_i to true if W^1_i contains x_i, and to false otherwise. Now, consider a clause c_i containing variables $x_{k_1}, x_{k_2}, x_{k_3}$. Because $a_i b_i$ appears only in snapshots $S^1_{k_1}, S^1_{k_2}, S^1_{k_3}$, and only in the component containing the literal that appears in c_i, we get that at least one of the three literals must be set to true.

Finally, we prove that the constraints can be assumed. First, we count the number of appearances. Observe that each edge of type $a_i b_i$ appears in exactly three snapshots, namely the snapshots related to the variables contained in c_i. Also, for a given variable x_i, the edges inside S^1_i between $x_i, \overline{x}_i$ and vertices in $\{a_j \mid j \in [m]\}$ appear once in that snapshot, and a second time in snapshot S^2_n. Finally, in a similar way the edges between T and $\{x_i, \overline{x}_i\}$ appear once in snapshot S^2_{i-1} if $i > 1$, once in snapshot S^2_i, and a third time in snapshot S^2_n. It follows that the problem is NP-complete even if each edge appears at most 3 times. Now, we want to spread snapshot S^2_n in a way that each snapshot becomes a forest with a constant number of edges. Observe that we could repeat the same pattern as the one in the first $2n - 1$ snapshots in order to visit the remaining edges in $G - \{a_j b_j \mid j \in [m]\}$. As long as the edges in $\{a_i b_i \mid i \in [m]\}$ appear only in the first $2n - 1$ snapshots, the same argument as before still applies, and we get the further constraint that each snapshot is a forest. Additionally, they can also be considered to have a constant number of edges since 3-SAT is NP-complete even if each variable appears at most three times [9]. Note that requiring both constraints to hold would increase the maximum number of appearances to 4.

If instead we are considering strictly increasing walks, a small modification of our construction will also imply NP-completeness, hence proving Corollary 2*(i)*. Indeed, it suffices to relate each variable x_i to a window big enough to ensure we will be able to visit all the edges of the considered component. Because each variable appears at most three times, one can see that it is enough that the edges are available for a period of 12 timestamps. So, our previous snapshot S_i^1 remains available for 12 consecutive timestamps, after which we will make S_i^2 available for 2 timestamps. Because the spare time can never be used to go from the component containing x_i to the component containing $\overline{x}_i$, the same arguments used in Theorem 5 still hold.

Now, if we consider a dynamic-based graph (G, λ), since all the edges are active throughout its lifetime, we clearly have that there exists an Eulerian walk if and only if G is connected; this proves the following Lemma.

Lemma 6 EULERIAN WALK *is polynomial for dynamic-based temporal graphs.*

By Lemma 4, Theorem 5, and Lemma 6, we obtain Item 1 of Theorem 1. Finally, note that if one is interested in closed walks instead, not only our NP-completeness reduction can be adapted in order to ensure that we can always go back to the initial vertex, but also the complexity results still hold.

4 Eulerian Local Tours and Trails

In this section we focus on Item 1 of Theorem 1. In the whole section, we will focus on dynamic-based temporal graphs as the hardness results for general temporal graphs are implied by the ones we prove for this restricted class. After the preliminary result in Lemma 7, we focus on proving the hardness result for the problem of deciding whether $(G, [2])$ has an Eulerian local tour, explaining the construction behind our reduction from NAE 3-SAT, whose correctness is proved in Theorem 8. We also argue that, if G is a cubic graph, then being Hamiltonian is a necessary but not sufficient condition for $(G, [2])$ to admit an Eulerian local tour, arguing the need of an *ad hoc* reduction for our problem. As the reduction in Theorem 8 focuses on solving EULERIAN LOCAL TOUR for $\tau = 2$, in Corollary 9 we extend this result to each fixed τ and to trails, thus completing the proof of Item 1 of Theorem 1. The following lemma helps us in our proof.

Lemma 7 *Let G be a graph. If $(G, [2])$ has an Eulerian local tour T, then T restricted to snapshot i must pass by all vertices of odd degree in G, for each $i \in [2]$.*

Proof For each $i \in [2]$, denote by T_i the trail in G equal to T restricted to timestamp i, and suppose, by contradiction, that $u \in V(G)$ is a vertex with odd degree not contained in T_1. Because T is a temporal tour, observe that T_1 is a trail in G starting at some s and finishing at some t, and T_2 is a trail in G starting at t and finishing at s, with possibly $s = t$. This means that the subgraph of G formed by the edges of T_2 is such that every $x \in V(G) \setminus \{s, t\}$ has even degree. This is a contradiction because, since no edge incident to u is visited in T_1, we get that all the edges incident to u must be visited in T_2, i.e., u would have odd degree in T_2. The same argument holds in case u is not in T_2, and the lemma follows.

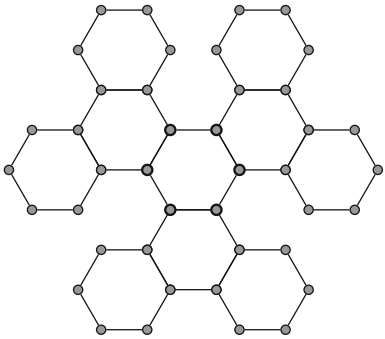


Fig. 2 Example of outerplanar graph G such that $(G, [2])$ does not have an Eulerian local tour

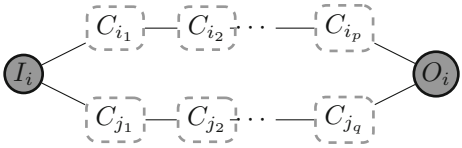


Fig. 3 Edge gadget with clause black boxes

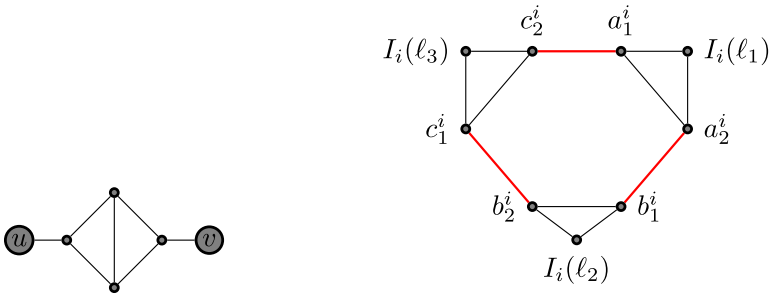
A simple consequence of the above lemma is that, as previously said, if G is cubic, then G must be Hamiltonian in order for $(G, [2])$ to have an Eulerian local tour. Since deciding whether a cubic graph is Hamiltonian is NP-complete [16], this hints towards the NP-completeness of the problem. However, since the other way around is not necessarily true (see e.g. the graph in Fig. 2), we need an explicit reduction. Indeed, the construction in Fig. 2 shows us that we might need an arbitrarily large lifetime in order to be able to visit all the edges of $(G, [\tau])$ even if G is a 2-connected outerplanar subcubic graph (which is trivially Hamiltonian).[2]

In the following we explain the construction behind our reduction from NAE 3-SAT. Let ϕ be a CNF formula on variables $\{x_1, \ldots, x_n\}$ and clauses $\{c_1, \ldots, c_m\}$. We start by presenting a meta-construction, in the sense that part of the constructed graph will be presented for now as black boxes and the actual construction is done later, as depicted in Fig. 3. The meta part concerns the clauses; so for now, denote by C_i the black box related to clause c_i. Without going into details, C_i will contain exactly one entry vertex for each of its literals, with some additional vertices, that will be presented later. So, given a literal ℓ contained in c_i, denote by $I_i(\ell)$ the entry vertex for ℓ in C_i. For each clause c_i with literals j, k, and ℓ, the three vertices $I_i(j)$, $I_i(k)$, and $I_i(\ell)$ are distinct.

Now, for each variable x_i, let $c_{i_1}, \ldots, c_{i_p}$ be the clauses containing x_i positively and $c_{j_1}, \ldots, c_{j_q}$ containing x_i negatively. Add two new vertices, I_i and O_i (these will be the entry and exit vertices for the variable gadget), and add the following edges (these compose the paths shown in Fig. 3):

$$\begin{aligned} E_i = &\{I_i I_{i_1}(x_i), I_i I_{j_1}(\overline{x}_i), I_{i_p}(x_i) O_i, I_{j_q}(\overline{x}_i) O_i\} \\ &\cup \{I_{i_h}(x_i) I_{i_{h+1}}(x_i) \mid h \in [p-1]\} \\ &\cup \{I_{j_h}(\overline{x}_i) I_{j_{h+1}}(\overline{x}_i) \mid h \in [q-1]\} \end{aligned}$$

[2] In order to get a cubic graph, as it is not necessary for the graph to be simple, it is possible to add multiple edges to the graph in Fig. 2 in order to make it cubic.



(a) Gadget related to a forced edge uv.

(b) Gadget related to clause c_i. Red edges represent forced edges.

Fig. 4 Gadgets for the reduction in Theorem 8

The paths will function as a switch, telling us whether the variable is true or false within the considered snapshot; we then denote by P_i the set of edges in the path $(I_i, I_{i_1}(x_i), \ldots, I_{i_p}(x_i), O_i)$, and by $\overline{P}_i$ the set of edges in the path $(I_i, I_{j_1}(\overline{x}_i), \ldots, I_{j_q}(\overline{x}_i), O_i)$. Now, to link the variable gadgets and to construct the clause gadgets, we will need a gadget that will function as an edge that must appear in the trail performed in G_1 and the one performed in G_2. For this, we use Lemma 7 applied to the gadget in Fig. 4a; when adding such a gadget between a pair u, v, we simply say that we are adding the *forced edge* uv.

Now, to link the variable gadgets, we add three new vertices s_1, s_2, t and the following forced edges.

$$E' = \{s_i t \mid i \in [2]\} \cup \{t I_1, O_n t\} \cup \{O_i I_{i+1} \mid i \in [n]\}.$$

The new vertices simply help us assume where the trail starts and finishes. Now, let T be an Eulerian local tour of $(G, [2])$ and denote by T_i the trail in G defined by T restricted to G_i, for $i \in [2]$. It is fairly easy to see (and we will prove it shortly) that if we can ensure that T_1 uses P_i if and only if T_2 uses $\overline{P}_i$, then we can prove equivalence with NAE 3-SAT. In other words, the clause gadget must be so that, for every clause c_j containing x_i (or equivalently $\overline{x}_i$), we get that either both edges incident to $I_j(x_i)$ in P_i (or equivalently $I_j(\overline{x}_i)$ in $\overline{P}_i$) are used, or none of them is used. Such a gadget is presented in Fig. 4b, where the red edges are forced.

Theorem 8 *Let G be a graph with degree at most 4. Then* Eulerian Local Tour *is NP-complete on* $(G, [2])$.

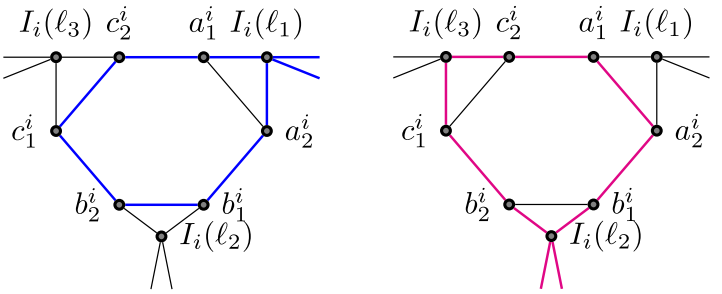
Proof Let ϕ and G be as previously constructed. First, consider a truth NAE assignment f to ϕ. We construct $T_1, T_2 \subseteq E(G)$ and prove that they form an Eulerian local tour of $(G, [2])$. Start by putting P_i in T_1 and $\overline{P}_i$ in T_2 if x_i is true, and the other way around if x_i is false. From now on, whenever we add a forced edge to T_1 and T_2, we are actually adding the trails depicted in Fig. 5.

Now, add E' to both T_1 and T_2, and consider c_i with literals ℓ_1, ℓ_2, ℓ_3. Suppose, without loss of generality, that ℓ_1 is true and ℓ_2 is false. We then add to T_1 the trail

(a) Trail added to T_1 when forced edge uv is added to T_1.

(b) Trail added to T_2 when forced edge uv is added to T_2.

Fig. 5 Trails related to forced edges



(a) Added to T_1 when ℓ_1 is true. **(b)** Added to T_2 when ℓ_2 is false.

Fig. 6 Trails in C_i related to a given NAE assignment

depicted in Fig. 6a, and to T_2 the one depicted in Fig. 6b. Observe that all internal edges of C_i are covered. Also note that the value of ℓ_3 is irrelevant (the choice remains the same, let it be true or false). We know that the remaining edges are also covered by $T_1 \cup T_2$ by construction. Finally, notice that both T_1 and T_2 touch all odd-degree vertices in a way that every vertex (including the even-degree ones) has even degree in T_1 and in T_2, except s_1, s_2 which have degree exactly 1. Also note that they form a connected graph; indeed they are formed by the cycle passing through the variable gadgets and t, together with some pending trails passing by the clause gadgets. Therefore, we can find an s_1, s_2-trail passing by all edges of T_1, and an s_2, s_1-trail passing by all edges of T_2, thus getting our Eulerian local tour.

For the reverse, let T be an Eulerian local tour of $(G, [2])$, and for each $i \in [2]$, denote by T_i the trail in G defined by T restricted to G_i. We need to prove that ϕ has an NAE satisfying assignment, and for this it suffices to show that, for every variable x_i, either P_i or $\overline{P}_i$ is contained in T_k for each $k \in [2]$. First observe that Lemma 7 indeed ensures that T_1 and T_2 restricted to the gadget related to a forced edge uv must be exactly as the trails depicted in Fig. 5; so in what follows we treat them exactly like edges that must appear in T_1 and T_2. Since there are 2 vertices of degree 1, namely s_1 and s_2, by Lemma 7 we can suppose that T_1 starts in s_1 and finishes in s_2, while T_2 starts in s_2 and finishes in s_1. Therefore, each of T_1 and T_2 contains a tour in $G - \{s_1, s_2\}$, and hence:

(I) Every vertex $u \in V(G) \setminus \{s_1, s_2\}$ has even degree in both T_1 and T_2.

Now, we prove that for each x_i, if T_1 intersects P_i, then T_1 does not intersect $\overline{P}_i$, the same holding for T_2. For this, consider c_i with literals ℓ_1, ℓ_2, ℓ_3, and for each $j \in [3]$, denote by $e_i^1(\ell_j)$, $e_i^2(\ell_j)$ the edges incident to $I_i(\ell_j)$ not contained in C_i. We first prove that, for each $j \in [3]$ and $k \in [2]$:

(II) Edge $e_i^1(\ell_j)$ is used in T_k if and only if edge $e_i^2(\ell_j)$ is used in T_k.

Without loss of generality, assume $j = 1$ and let $k \in [2]$; suppose by contradiction that $e_i^1(\ell_1)$ is in T_k, while $e_i^2(\ell_1)$ is not in T_k. By (I) we get that either $a_1^i I_i(\ell_1)$ or $a_2^i I_i(\ell_1)$ is in T_k, say $a_1^i I_i(\ell_1)$. But then we get that $a_1^i a_2^i$ and $a_2^i I_i(\ell_1)$ are not in T_k, as otherwise either a_1^i or $I_i(\ell_1)$ would have odd degree in T_k. This is a contradiction since we then get a_2^i with degree 1 in T_k. The same argument can be analogously applied when $a_2^i I_i(\ell_1)$ is in T_k or when $j \in \{2, 3\}$.

Consider now a variable x_i, and let $c_{i_1}, \ldots, c_{i_p}$ be the clauses containing x_i positively and $c_{j_1}, \ldots, c_{j_q}$ containing x_i negatively. Because I_i has degree 3 in G and $E' \setminus \{s_i t \mid i \in [2]\}$ is contained in $T_1 \cap T_2$, we get that exactly one edge between $I_1 I_{i_1}(x_i)$ and $I_1 I_{j_1}(\overline{x}_i)$ is contained in T_1. From (II), we then get that either P_i is contained in T_1 or $\overline{P}_i$ is contained in T_1. Observe that this implies that P_i is contained in T_k, while $\overline{P}_i$ is contained in T_{3-k}, for some $k \in [2]$. We then set x_i to be true if and only if T_1 contains P_i. Because the edges in $E_i = \{e_i^1(\ell_j), e_i^2(\ell_j) \mid j \in [3]\}$ separate C_i from the rest of the graph and by (I), we get that both T_1 and T_2 must intersect E_i. Finally by (II) we get that this assignment is a NAE truth assignment for ϕ. □

Observe that if we add two new vertices of degree one adjacent to the vertex t, then we get a reduction to the problem of deciding whether the edges of G can be covered by two trails, proving Corollary 3. The following corollary concludes the proof of Item 1 in Theorem 1.

Corollary 9 Eulerian Local Tour *and* Eulerian Local Trail *are NP-complete on temporal graphs with lifetime τ for every fixed $\tau \geq 2$. This also holds on dynamic-based graphs.*

Proof We first make a reduction from Eulerian Local Tour on $(G, [2])$ to Eulerian Local Trail on $(G', [\tau])$. Given $(G, [2])$ constructed as in the proof of Theorem 8, let G' be obtained from G by adding a star on $\tau+2$ vertices and identifying one of its leaves with s_1. We argue that $(G, [2])$ has an Eulerian local tour (starting and finishing in s_1) if and only if $(G', [\tau])$ has an Eulerian local trail. Denote the vertices of the initial star by $u, v_1, \ldots, v_{\tau+1}$, where u is the central vertex, and $v_2 = s_1$ is the vertex where G is pending. Let T be an Eulerian local tour of $(G, [2])$ starting and finishing in s_1, and T_1, T_2 be the trails in G defined by T. Build an Eulerian local trail of $(G', [\tau])$ by visiting $v_1 u$, $u v_2$ and T_1 in G'_1, then performing T_2 and visiting $v_2 u$ and $u v_3$ in G'_2, and finish visiting the remaining edges of the star in the obvious way.

Now, let T be an Eulerian local trail of $(G', [\tau])$, and denote by T_i the trail in G' defined by T restricted to G'_i, for each $i \in [\tau]$. Observe that because we have $\tau + 1$ cut edges, we get that each T_i contains at most 2 of them, and in case it contains exactly 2, say $v_1 u, u v_2$, then T_{i+1} either does not contain any cut edge, or must intersect T_i in $v_1 u, u v_2$. This means that the best we can do in order to finish by time τ is to visit exactly two of them in the first snapshot, and exactly one more in each of the

subsequent snapshots. We can therefore suppose, without loss of generality that T_i contains $v_i u$, $v_{i+1} u$ for each $i \in [\tau]$. Note that this implies that every edge of $(G, [2])$ must be visited in T_1 and T_2, with T_1 starting in v_2 and T_2 finishing in v_2, as we wanted to prove.

Finally, note that $(G', [\tau])$ constructed above has an Eulerian local trail if and only if $(G', [\tau + 1])$ has an Eulerian local tour. This completes our proof. □

Finally, in order to prove Corollary 2.(ii), which considers strictly increasing local trails, we can make a modification similar to the one made for walks. Observe that this transformation results in a non-dynamic-based temporal graph.

Proof of Corollary 2.(ii) As made for walks, we slice the lifetime of (G, λ) into windows, each window allowing only for the edges of a given variable to appear. For this, first observe that, for each clause c_j, a pass inside of C_j uses at most 20 edges inside of C_j (already considering that a pass through a forced edge consists of a path on 5 edges). This means that if we allow the edges of a variable x_i appearing in c_j to live long enough, we will be able to visit C_j in a strictly increasing way. For simplicity, consider again that each variable appears at most 3 times in ϕ. We assign to each x_i two time windows, one for the first passing, one for the second, each of size 70. Thus, variable x_1 will take windows $\{1, \ldots, 70\}$ and $\{70n+1, \ldots, 70(n+1)\}$, with the edges of $P_1 \cup \overline{P}_1 \cup \{O_1 I_2\}$ being active in the following times: the first edge of P_1 and of $\overline{P}_1$ are active in time 1 and $70n+1$, the last edges of P_1 and of $\overline{P}_1$ are active in time 65 and $70n+65$, forced edge O_1, I_2 is active in times $\{66, \ldots, 70, 70n+66, \ldots, 70(n+1)\}$, and the remaining edges are active in the period $\{2, \ldots, 64\} \cup \{70n+2, \ldots, 70n+64\}$. Similarly the window of x_i will be $\{70(i-1), \ldots, 70i\} \cup \{70(n+i-1), \ldots, 70(n+i)\}$. Note that we can link O_n to I_1 trough a direct edge, making it active in time $\{70n\}$; this can be combined with the end of the first window of x_n since in this window the last edges of P_n and $\overline{P_n}$ are active in time $70n - 5$. Finally, the edges inside a clause c_j will be active during the windows of the corresponding literals. One can verify that the key Property (II) in the proof of Theorem 8 holds, and NP-completeness follows. □

5 Eulerian Tours and Trails

We finally focus on Eulerian Trail and Eulerian Tour, proving that in the general case they are both NP-complete, hence, proving Item 1 in Theorem 1. To this aim, we make an adaptation of the construction in Theorem 8. Observe that here the base graph needs to be Eulerian as otherwise the answer to Eulerian Trail is trivially NO. This also implies that the problem restricted to dynamic-based graphs is trivial: if the base graph is Eulerian, then the answer to Eulerian Tour is YES; otherwise, then the answer is NO. The trick now is to take advantage of the function λ in order to enforce the edges.

Theorem 10 Eulerian Tour *and* Eulerian Trail *are NP-complete, even on temporal graphs with fixed lifetime* $\tau \geq 2$.

Proof We first prove the case $\tau = 2$. For this, we simply replace the gadget to enforce an edge uv in the construction of Sect. 4 by two paths of length 2, P^1_{uv} and P^2_{uv},

where the edges in P^i_{uv} are active only in snapshot G_i, for each $i \in [2]$. Because the arguments used in Sect. 4 depended only on the fact of uv indeed being an enforced edge, we can apply the same arguments here. The only difference is that the trails in G_1 and G_2 now cannot intersect, which indeed is the case since the intersection between T_1 and T_2 in Sect. 4 is exactly the set of forced edges, and since here each appearance of a forced edge uv is actually related either to P^1_{uv} or to P^2_{uv}.

Now, in order to prove the NP-completeness for higher values of τ, we add new vertices $v_3, \ldots, v_\tau$ and edges $\{s_2v_3\} \cup \{v_iv_{i+1} \mid i \in \{3, \ldots, \tau-1\}\}$, where $\lambda(s_2v_3) = \{3\}$ and $\lambda(v_iv_{i+1}) = \{i+1\}$ for each $i \in [\tau-1]$. This gives us that Eulerian Trail is NP-complete on (G, λ) with lifetime τ for every fixed $\tau \geq 2$. And if we want a closed trail, it suffices to identify v_τ with s_2, if $\tau \geq 4$, and if $\tau = 3$, we add a new vertex v_4 and edges v_3v_4, v_4s_2 active only in snapshot G_3. This concludes our proof.

Again, in order to prove Corollary 2.(iii), a modification similar to the one made for walks works. We give a more formal argument below.

Proof of Corollary 2.(iii) As previously said, we will slice the lifetime of (G, λ) into windows, each window allowing only for the edges of a given variable to appear. For this, first observe that, for each clause c_j, a pass inside of C_j uses at most 11 edges (already considering that a forced edge is being replaced by two paths on 2 edges). This means that if we allow the edges of a variable x_i appearing in c_j to live long enough, we will be able to visit C_j in a strictly increasing way. For simplicity, consider again that each variable appears at most 3 times in ϕ. We assign to each x_i two time windows, one for the first passing, one for the second, each of size 40 (could be 39, but we choose that for roundness). Thus, variable x_1 will take windows $\{1, \ldots, 40\}$ and $\{40n+1, \ldots, 40(n+1)\}$, with the edges of $P_1 \cup \overline{P}_1 \cup \{O_1I_2\}$ being active in the following times: the first edge of P_1 and of $\overline{P}_1$ are active in time 1 and $40n+1$, the last edges of P_1 and of $\overline{P}_1$ are active in time 38 and $40n+38$, forced edge O_1, I_2 is active in times $\{39, 40, 40n+39, 40(n+1)\}$, and the remaining edges are active in the period $\{2, \ldots, 37\} \cup \{40n+2, \ldots, 40n+37\}$. Similarly the window of x_i will be $\{40(i-1), \ldots, 40i\} \cup \{40(n+i-1), \ldots, 40(n+i)\}$. Note that we can link O_n to I_1 directly, making them active during $\{40n-1, 40n\}$ (this is the end of the first window of x_n). Finally, the edges inside a clause c_j will be active during the windows of the corresponding literals. One can verify that the key Property (II) in the proof of Theorem 8 holds, and NP-completeness follows.

6 Underlying Graph with Bounded Treewidth

In this section we study our problems when applied to temporal graphs (G, λ), where the graph G has bounded treewidth. In this case, first we show some negative results which are consequences of the results by [1, 5], thus proving the negative blue results in Table 1. We remark that these negative results, contrarily to the ones in the previous sections, require the lifetime τ being unbounded. In the remainder of the section we complete the picture giving new algorithms for Eulerian Trail, Eulerian Local Trail and the corresponding tour problems that run in XP time when parameterized by $\tau + tw(G)$. Observe that the reductions in previous sections, combined with the

following ones, exclude the possibility of algorithms XP when parameterized only by one of these two parameters.

Theorem 11 *Given a temporal graph (G, λ) the following results hold.*

1. Strict Eulerian Walk *is NP-complete when G is a star* [1, 5]. Eulerian Walk *is NP-complete when G is a tree.*
2. Strict Eulerian Local Trail *and* Eulerian Local Trail *are NP-complete when G is a star* [1, 5].
3. Strict Eulerian Trail *and* Eulerian Trail *are trivially polynomial for trees. And if $tw(G) = 2$, then* Strict Eulerian Trail *is NP-complete* [5]*, as well as* Eulerian Trail.

In order to prove Theorem 11, some preliminary results on the so-called TEXP problem introduced by [25] are needed. TEXP consists of, given a temporal graph (G, λ), finding a temporal walk that visits all vertices in G (possibly, more than once) whose arrival time is minimum. In the case in which just strictly increasing walks are allowed, [1] proved that TEXP is NP-complete on temporal stars in which each edge appears at most k times, for all fixed $k \geq 6$, and this result has been then improved by [5] for all fixed $k \geq 4$. Since in a star, passing by all the leaves translates also into passing by all the edges, their result implies already NP-completeness for Strict Eulerian Walk for stars and hence for trees. However, in the case of Eulerian Walk these reductions cannot be applied directly and some modifications are needed, as shown by the following lemma.

Lemma 12 *When increasing walks (not necessarily strictly increasing) are allowed, TEXP is NP-complete on trees even if each edge appears at most four times.*

Proof We modify the reduction in [5, Theorem 1], noting that their reduction does not work directly when not necessarily strictly increasing walks are allowed. Indeed, when traversing an edge now we are allowed to go back on the same edge at the same snapshot (possibly entering more edges than expected), while the reduction requires that when traversing an edge we will go back to the center of the star in a subsequent different time. We modify the reduction, adding new leaves to the star, which will be available at suitable times in order to maintain the requirement. Doing this, our graph is not a star anymore but it is still a tree.

We modify the reduction following its notation, where, for each edge e, $\tau(e)$ corresponds to our $\lambda(e)$. We do the following changes:

- In the original reduction multiply all the times by 2, i.e. for each edge e each value in $\tau(e)$ becomes the double.
- For any i, add the edges u_i, each one adjacent only to the edge e_i.
- For any ζ, j, k, add the edges u^{ζ}_{jk}, each one adjacent only to e^{ζ}_{jk}.
- For each e_i, let $\tau(e_i) = \{t^i_0, t^i_1, t^i_2, t^i_3\}$ be the times in strictly increasing order as specified by the original reduction. [5, Equation 2] and multiplied by 2. Then $\tau(u_i) = \{\frac{t^i_0+t^i_1}{2}, \frac{t^i_1+t^i_2}{2}, \frac{t^i_2+t^i_3}{2}\}$.
- For each e^{ζ}_{jk}, let $\tau(e^{\zeta}_{jk}) = \{r_1, r_2, r_3, r_4\}$ be the times in strictly increasing order as specified by the original reduction [5, Equation 3] and multiplied by 2. Then $\tau(u^{\zeta}_{jk}) = \{\frac{r_1+r_2}{2}, \frac{r_2+r_3}{2}, \frac{r_3+r_4}{2}\}$.

We have thus obtained a temporal graph whose underlying graph is formed by a center with paths of length two (formed by edges e_i and u_i or by edges e^{ζ}_{jk} and u^{ζ}_{jk}) departing from it, where each edge has still at most four temporal occurrences. Now, if entering from the center in e_i at time t^i_x with $x \in \{0, 1, 2\}$, it is not possible to go back at time t^i_x because also u_i must be visited and the first time in which it will be available is $\frac{t^i_x + t^i_{x+1}}{2}$ which is strictly greater than t^i_x.

Proof of Theorem 11 1. On stars STRICT EULERIAN WALK is equivalent to TEXP, which has been proved to be NP-complete by [1, 5]. The NP-completeness for EULERIAN WALK in the case of trees follows from Lemma 12.
2. Since a strict walk is also a strict local trail and vice versa, by Item 1 we obtain that STRICT EULERIAN LOCAL TRAIL is also NP-complete when G is a star. Additionally, observe that the argument also holds if we constrain to local trails instead. Indeed every local trail is a strict walk, since it is not allowed for an edge to be visited twice in the same snapshot, and every strict walk must be a local trail similarly.
3. As for STRICT EULERIAN TRAIL and EULERIAN TRAIL we cannot repeat edges of G. In the case of trees, these problems are trivially polynomial-time solvable, as it suffices to check whether the graph is connected and is a path. The fact that if $tw(G) = 2$ then STRICT EULERIAN TRAIL is NP-complete has been proved by [5] and by simply applying the idea of their reduction to the trees in Lemma 12 this result extends to EULERIAN TRAIL. Indeed, given a temporal graph (G, λ) where G is a tree like the one in Lemma 12, i.e. a center c with s paths formed by edges e_i, u_i (with $i \in \{1, \ldots, s\}$), where $\lambda(e_i) = \{t_1, t_3, t_5, t_7\}$ and $\lambda(u_i) = \{t_2, t_4, t_6\}$ (with $t_i < t_j$ if $i < j$), (G, λ) can be transformed into (G', λ') such that there is a (not necessarily strictly increasing) temporal exploration in (G, λ) iff there is a (not necessarily strictly increasing) Eulerian trail in (G', λ'). G' is such that each path e_i, u_i becomes a cycle formed by the edges $e_i, x_i, u_i, x'_i, e'_i$ (see the right of Figure 2 by [5] and replace each edge incident to c by a path of two edges). Then λ' is as follows: $\lambda'(e_i) = \{2t_1, 2t_3, 2t_5\}$, $\lambda'(x_i) = \{t_1 + t_2, t_3 + t_4, t_5 + t_6\}$, $\lambda'(u_i) = \{2t_2, 2t_4, 2t_6\}$, $\lambda'(x'_i) = \{t_2 + t_3, t_4 + t_5, t_6 + t_7\}$, $\lambda'(e'_i) = \{2t_3, 2t_5, 2t_7\}$.

6.1 An XP Algorithm for Local Trails and Trails

In the following, we present an algorithm for EULERIAN LOCAL TRAIL and EULERIAN TRAIL which runs in XP time when parameterized by the treewidth and the lifetime. It easily adapts to the tour versions of the problems, thus proving the following theorem.

Theorem 13 *Let $\mathcal{G} = (G, \lambda)$ be a temporal graph with lifetime τ, and such that $tw(G) = k$ and $|V(G)| = n$. Then* EULERIAN LOCAL TRAIL, EULERIAN LOCAL TOUR, EULERIAN TRAIL *and* EULERIAN TOUR *can be solved in $\mathcal{G}$ in time $2^{k^2 \cdot \tau} \cdot n^{\tau}$.*

We start by solving the EULERIAN LOCAL TRAIL and later explain what needs to be changed in order to solve EULERIAN TRAIL. Before we start, we give the definitions related to tree decomposition.

Treewidth notions and notations

Given a graph G, a *tree decomposition of* G is a pair $\mathcal{T} = (T, \mathcal{X})$ where T is a tree, and $\mathcal{X}$ assigns to each $t \in V(T)$ a subset of $X_t \subseteq V(G)$ such that:

1. $\bigcup_{t \in V(T)} X_t = V(G)$;
2. For every $uv \in E(G)$, there exists $t \in V(T)$ such that $\{u, v\} \subseteq X_t$; and
3. For every $t, t' \in V(T)$ and every t'' in the t, t'-path in T, we have that $X_t \cap X_{t'} \subseteq X_{t''}$.

The *treewidth of* $\mathcal{T}$ is equal to $\max_{t \in V(T)} |X_t| - 1$, while the *treewidth of* G is the minimum treewith over all all tree decompositions of G; it is denoted by $tw(G)$. To avoid confusion, the vertices of T are referred to as *nodes*. Also, the subsets in $\mathcal{X}$ are called *bags*. As usually done, we make use of a more well-behaved tree decomposition. A *nice tree decomposition* is a tree decomposition $(T, \mathcal{X})$ such that T is rooted in a vertex r, and each $t \in V(T)$ is one of the following types of nodes:

- *Leaf node*: in this case, t is a leaf of T and X_t is empty;
- *Forget node*: t has exactly one child, t', and $X_t = X_{t'} \setminus \{v\}$;
- *Introduce node*: t has exactly one child, t', and $X_t = X_{t'} \cup \{v\}$;
- *Join node*: t has exactly two children, t_1, t_2, and $X_t = X_{t_1} = X_{t_2}$.

It is known that a nice tree decomposition of width at most $5tw(G)$ can be computed in time $2^{O(tw)}$ [3, 20].

Given a nice tree decomposition $(T, \mathcal{X})$, with T rooted in r, and a node t of T, we denote by T_t the subtree of T rooted in t, by V_t the set containing all the vertices in a bag of T_t (i.e., $V_t = \{u \in V(G) \mid u \in X_{t'} \text{ for some } t' \in V(T_t)\}$, by G_t the subgraph $G[V_t]$, and by $E(X_t)$ the set of edges in $G_t[X_t]$. To avoid confusion with the snapshots, here we will be denoting the i-th snapshot by $G[i]$.

An Auxiliary Problem: Partial Eulerian Local Trail

Now, given a temporal graph $\mathcal{G} = (G, \lambda)$ with lifetime τ, we work on a nice tree decomposition $(T, \mathcal{X})$ of G in a bottom-up way, i.e., we solve the problem first on the leaves and move up to the root, assuming, when computing an entry of the table of a node t, that the tables of its children are known. We solve the following problem instead, which we call Partial Eulerian Local Trail: given $u_1, \ldots, u_{\tau+1} \subseteq V(G)$, decide whether $\mathcal{G}$ has a temporal Eulerian local trail passing by $(u_1, \ldots, u_{\tau+1})$. In other words, we decide whether there are subgraphs $W_1, \ldots, W_\tau$ of $G[1], \ldots, G[\tau]$, respectively, such that $\bigcup_{i=1}^{T} E(W_i) = E(G)$ and, for every $i \in [\tau]$, we have that W_i is connected, every $u \in V(W_i) \setminus \{u_i, u_{i+1}\}$ has even degree in W_i, and either $u_i \neq u_{i+1}$ and they have odd degree in W_i, or $u_i = u_{i+1}$ and it has even degree in W_i. Clearly, if this problem can be solved in time $f(tw(G), \tau)n^{O(1)}$, then Eulerian Local Trail can be solved in time $f(tw(G), \tau)n^{\tau}$.

The Algorithm

To solve Partial Eulerian Local Trail, we keep partial solutions of G_t, for each node t of a nice tree decomposition, which means that these partial solutions might be disconnected as well as have more vertices with odd degree than allowed. In what follows, we describe the information that will be kept in each node, supposing that a partial solution $W_1, \ldots, W_\tau$ for G_t is known:

- $X^i \subseteq X_t$: this is the subset of X_t touched by W_i, i.e. $X^i = V(W_i) \cap X_t$;
- A partition $\mathcal{C}_i$ of X^i: this keeps track of the subsets of vertices of X^i contained in the same connected component of W_i. More formally, if $C_1, \ldots, C_q$ are all the connected components of W_i, then $\mathcal{C}_i = \{V(C_j) \cap X_t \mid j \in [q]\}$;
- A characteristic vector p_i of length $|X_t|$: this keeps track of the parity of the degree of each vertex in the partial solution. More formally, for each $u \in X_t$, $p_i(u) = d_{W_i}(u) \mod 2$; and finally
- A characteristic vector e_i of length $|E(X_t)|$: this keeps track of which edges from $E(X_t)$ appear in W_i. More formally, $e_i(uv) = 1$ if and only if $uv \in E(W_i)$.

The above description is given so as the reader can follow the algorithm more easily. However, we clearly cannot keep all the partial solutions as otherwise our algorithm cannot work in the desired time. What is done instead is that we keep only the parameters defined above and then apply dynamic programming in order to compute the desired solution.

Given a node $t \in V(T)$, we define the table $\mathcal{E}_t$ indexed by $(X^i, \mathcal{C}_i, p_i, e_i)_{i\in[\tau]}$, and we say that $\mathcal{E}_t((X^i, \mathcal{C}_i, p_i, e_i)_{i\in[\tau]}) = 1$ if and only if there exist $W_1, \ldots, W_\tau$ such that $W_i \subseteq G_t[i]$ for every $i \in [\tau]$, and the following hold:

1. $\bigcup_{i=1}^{\tau} E(W_i) = E(G_t)$;
2. $V(W_i) \cap X_t = X^i$, for every $i \in [\tau]$;
3. If $C_1, \ldots, C_q$ are the connected components of W_i, then $\mathcal{C}_i = \{V(C_j) \cap X_t \mid j \in [q]\}$, for every $i \in [\tau]$. Furthermore, either $V(C_j) \cap X_t \neq \emptyset$ for every $j \in [q]$, or $q = 1$ and $\{u_i, u_{i+1}\} \subseteq V(C_1)$;
4. For every $i \in [\tau]$, we have:

$$d_{W_i}(u) \bmod 2 = \begin{cases} p_i(u), & \text{if } u \in X_t \\ 0, & \text{if } u \in V(G_t) - X_t - \{u_i, u_{i+1}\} \\ 0, & \text{if } u \in (V(G_t) - X_t) \cap \{u_i, u_{i+1}\} \\ & \text{and } u_i = u_{i+1} \\ 1, & \text{if } u \in (V(G_t) - X_t) \cap \{u_i, u_{i+1}\} \\ & \text{and } u_i \neq u_{i+1} \end{cases}$$

5. $e_i(uv) = 1$ if and only if $uv \in E(W_i)$, for each $i \in [\tau]$ and each $uv \in E(X_t)$.

First, we show that if the tables are known, then we can find the answer to Partial Eulerian Local Trail in $\mathcal{E}_r$, where r is the root of the tree in the tree decomposition.

Lemma 14 *There exists a temporal Eulerian local trail passing by $(u_1, \ldots, u_{\tau+1})$ if and only if $\mathcal{E}_r((X^i, \mathcal{C}_i, p_i, e_i)_{i\in[\tau]}) = 1$, for some tuple $((X^i, \mathcal{C}_i, p_i, e_i)_{i\in[\tau]})$ such that, for every $i \in [\tau]$: if $X^i \neq \emptyset$, then $\mathcal{C}_i = \{X^i\}$; and $p_i(u) = 0$, for every $u \in X_t \setminus \{u_i, u_{i+1}\}$, $p_i(u) = 0$ if $u \in X_t \cap \{u_i, u_{i+1}\}$ and $u_i = u_{i+1}$, and $p_i(u) = 1$ if $u \in X_t \cap \{u_i, u_{i+1}\}$ and $u_i \neq u_{i+1}$.*

Proof First, consider a temporal Eulerian local trail $\mathcal{W} = (W_1, \ldots, W_\tau)$ where W_i is a u_i, u_{i+1}-trail in $G[i]$, for every $i \in [\tau]$. Then build the tuples $(X^i, \mathcal{C}_i, p_i, e_i)$ accordingly, i.e., for each $i \in [\tau]$, define:

- $X^i = V(W_i) \cap X_r$;

- $\mathcal{C}_i = \{X^i\}$, if $X^i \neq \emptyset$. Otherwise, $\mathcal{C}_i = \emptyset$;
- Define p_i exactly as in the statement of the theorem; and finally
- For each $uv \in E(X_r)$, let $e_i(uv) = 1$ if and only if $uv \in E(W_i)$.

We need to prove that $\mathcal{E}_r((X^i, \mathcal{C}_i, p_i, e_i)_{i\in[\tau]}) = 1$, i.e., that Conditions 6.1–6.1 hold. First, recall that $G_r = G$; hence Condition 6.1 holds because $\mathcal{W}$ is Eulerian. Conditions 6.1 and 6.1 hold by construction. Additionally, we know that since each W_i is a u_i, u_{i+1}-trail in G, we get that $d_{W_i}(u)$ is even for every $u \notin \{u_i, u_{i+1}\}$, and that $d_{W_i}(u_i)$ and $d_{W_i}(u_{i+1})$ are either both odd when $u_i \neq u_{i+1}$, or $u_i = u_{i+1}$ and $d_{W_i}(u_i)$ is even. This and the construction of p_i ensure Condition 6.1. Finally, because W_i is connected, we know that either W_i does not pass through X_r and hence $X^i = \mathcal{C}_i = \emptyset$, or W_i intersects X_r and $\mathcal{C}_i = \{V(W_i) \cap X_r\} = \{X^i\}$. This ensures Condition 6.1.

Now, suppose that $\mathcal{E}_r((X^i, \mathcal{C}_i, p_i, e_i)_{i\in[\tau]}) = 1$ for some entry satisfying the statement, and let $W_1, \ldots, W_\tau$ be subgraphs satisfying Conditions 6.1–6.1. If it holds that each W_i is connected and such that every vertex has even degree, except u_i and u_{i+1} when they are distinct vertices, then by Condition 6.1 and Euler's characterization we get that $\mathcal{G}$ has a temporal Eulerian local trail, as desired. Therefore, consider $i \in [\tau]$. If $X^i \neq \emptyset$, then by assumption we know that $\mathcal{C}_i = \{X^i\}$, and by Condition 6.1 we get that W_i contains exactly one component. And if $X^i = \emptyset$, then by Condition 6.1 we get $V(W_i) \cap X_r = \emptyset$, which in turn implies that $V(C) \cap X_r = \emptyset$ for every component C of W_i, which again by Condition 6.1 implies that W_i is connected. Now, concerning the parity of the degrees, by the lemma hypothesis concerning p and by Condition 6.1, one can see that indeed the only vertices allowed to have odd degrees are u_i and u_{i+1}, and only if they are distinct. □

In what follows, we explain how to compute the entry of each type of node, given that the tables of the children are known. We prove correctness for each of the operations of the nice tree decomposition through the following lemmas. We refrain from formally proving correctness of the whole algorithm since these ideas follow standard techniques.

Introduce Node

Consider an entry $\mathcal{E}_t((X^i, \mathcal{C}_i, p_i, e_i)_{i\in[\tau]})$ that we want to compute, and first consider t to be an introduce node with child t', where $X_t = X_{t'} \cup \{u\}$. We explain which entries of $\mathcal{E}_{t'}$ must be searched for the right answer. First, observe that if $u \notin X^i$, then nothing changes concerning $G_t[i]$, i.e., there will be the desired u_i, u_{i+1}-trail in $G_t[i]$ if and only if such trail also exists in $G_{t'}[i]$. So suppose $u \in X^i$ and let $C \in \mathcal{C}_i$ be such that $u \in C$; also denote $X^i - u$ by Y^i. Recall that $G_{t'} = G_t - u$ and note that no changes can be made to the edges of $W_i - u$; hence let e'_i denote the vector e_i restricted to $E(X_{t'})$. Observe also that the parity of the degree of $v \in X_{t'}$ in W_i restricted to $G_{t'}$ might differ from its degree in W_i depending on whether v is adjacent to u in W_i or not, which can be checked by looking at e_i. Hence, we construct a new parity vector p':

$$p'_i(v) = \begin{cases} p_i(v) & , \text{if } v \notin N(u) \text{ or } e_i(uv) = 0 \\ (p_i(v) + 1) \mod 2 & , \text{if } v \in N(u) \text{ and } e_i(uv) = 1 \end{cases}$$

Finally, note that u might be the vertex in $W_i \subseteq G_t$ connecting some of the vertices of C. Therefore, we need to consider the partition $\mathcal{C}'_i$ obtained from $\mathcal{C}_i$ by removing u from C, but also all the partitions obtained from $\mathcal{C}'_i$ by partitioning also $C - u$ into further subsets. Let χ_i denote the set of all such partitions. We then get that:

Lemma 15 *Consider an introduce node t with child t', $u \in V(G)$ be such that $X_t = X_{t'} \cup \{u\}$, and consider an entry $(X^i, \mathcal{C}_i, p_i, e_i)_{i\in[\tau]}$ of $\mathcal{E}_t$. If $u \notin X^i$, then $\mathcal{E}_t((X^i, \mathcal{C}_i, p_i, e_i)_{i\in[\tau]}) = 1$ if and only if $\mathcal{E}_{t'}((X^i, \mathcal{C}_i, p_i, e_i)_{i\in[\tau]}) = 1$. Otherwise, let Y^i, p'_i, e'_i, χ_i be constructed as before, for each $i \in [\tau]$. Then, $\mathcal{E}_t((X^i, \mathcal{C}_i, p_i, e_i)_{i\in[\tau]}) = 1$ if and only if there exist $\mathcal{D}_1, \ldots, \mathcal{D}_\tau$ such that $\mathcal{D}_i \in \chi_i$ for every $i \in [\tau]$, and $\mathcal{E}_{t'}((Y^i, \mathcal{D}_i, p'_i, e'_i)_{i\in[\tau]}) = 1$.*

Note that each χ_i has size at most one plus the total number of partitions of $C - u \subseteq X_{t'}$, which is at most k^k, where $k = tw(G)$. Because we need to investigate all the possible combinations of $\mathcal{D}_1, \ldots, \mathcal{D}_\tau$, we get that at most $k^{k\cdot\tau}$ entries of $\mathcal{E}_{t'}$ must be accessed.

Forget Node

Now, consider t to be a forget node with child t', where $X_t = X_{t'} \setminus \{u\}$. The vertices of $X_{t'}$ touched by W_i must be the same, except that perhaps u can be touched too. And in fact, if $u \in \{u_i, u_{i+1}\}$, then u must appear in W_i. Therefore, define $\mathcal{X}^i$ to be equal to $\{X^i, X^i \cup \{u\}\}$ if $u \notin \{u_i, u_{i+1}\}$; otherwise, let $\mathcal{X}^i$ denote $\{X^i \cup \{u\}\}$. As for the partition of $X \in \mathcal{X}^i$ to be considered, since $G_t = G_{t'}$, it cannot change with respect to $\mathcal{C}_i$ unless $u \in X$, in which case there is a family of partitions that must be considered. Define then the family of partitions χ_i containing each $\mathcal{C}'_i$ that can be obtained from $\mathcal{C}_i$ by adding u to some part, including the possibility of u being contained in a part by itself. Now consider $e'_i \in \{0, 1\}^{|E(X_{t'})|}$; we say that e'_i is an *extension of* e_i if $e'_i(vw) = e_i(vw)$ for every $vw \in E(X_t)$ (i.e., e'_i differs from e_i only on edges incident to u). Consider also $\mathcal{D}_i \in \chi_i$. If $\{u\} \in \mathcal{D}_i$, then uv cannot be in W_i for every $uv \in E(X_t)$. And if $u \in C$ for some $C \in \mathcal{D}_i$, then uv can be in W_i for some $uv \in E(X_t)$ only if $v \in C$. We then say that e'_i *agrees with* $\mathcal{D}_i$ if these conditions hold, i.e., if $v \in C$ whenever $e'_i(uv) = 1$, where $C \in \mathcal{D}_i$ is the part containing u. Finally, because $G_t = G_{t'}$, the parities of the degrees do not change; also, since $u \in V(G_t - X_t)$, we get that either u must have even degree in W_i, or $u \in \{u_i, u_{i+1}\}$ and $u_i \neq u_{i+1}$, in which case u must have odd degree in W_i. We then define p'_i to be obtained from p_i by setting $p'_i(u)$ to 0 if $u \notin \{u_i, u_{i+1}\}$ or $u_i = u_{i+1}$, or to 1 otherwise. These definitions lead us to the next lemma.

Lemma 16 *Consider a forget node t with child t', $u \in V(G)$ be such that $X_t = X_{t'} \setminus \{u\}$, and consider an entry $(X^i, \mathcal{C}_i, p_i, e_i)_{i\in[\tau]}$ of $\mathcal{E}_t$. Then, $\mathcal{E}_t((X^i, \mathcal{C}_i, p_i, e_i)_{i\in[\tau]}) = 1$ if and only if either $u \notin \{u_i, u_{i+1}\}$ and $\mathcal{E}_{t'}((X^i, \mathcal{C}_i, p_i, e_i)_{i\in[\tau]}) = 1$, or there exists $(Y^i, \mathcal{D}_i, p'_i, e'_i)_{i\in[\tau]}$ such that, for each $i \in [\tau]$, we have that $Y^i \in \mathcal{X}^i$, $\mathcal{D}_i \in \chi_i$, e'_i is an extension of e_i that agrees with $\mathcal{D}_i$, and $\mathcal{E}_{t'}((Y^i, \mathcal{D}_i, p'_i, e'_i)_{i\in[\tau]}) = 1$.*

Again, let $k = tw(G)$. Observe that $|\chi_i| \leq |\mathcal{C}_i| + 1 \leq k + 1$, and that there are at most 2^k possible extensions of e_i. Because $|\mathcal{X}^i| \leq 2$, and all combinations must be investigated, we get a total of at most $(2^{k+1}(k + 1))^\tau$ entries that need to be checked.

Join Node

Finally, consider a join node t with children t_1, t_2, and recall that $X_t = X_{t_1} = X_{t_2}$. Given partitions $\mathcal{P}_1, \mathcal{P}_2$ of a set X, we say that a partition $\mathcal{P}$ of X is the *union* of $\mathcal{P}_1$ and $\mathcal{P}_2$ if it is equal to the equivalent classes of the transtive closure of $\{(u, v) \in X \times X \mid \{u, v\} \subseteq C$, for some $C \in \mathcal{P}_1 \cup \mathcal{P}_2\}$. Given entries $h_1 = (X^i, \mathcal{C}_i^1, p_i^1, e_i)_{i\in[\tau]}$ and $h_2 = (X^i, \mathcal{C}_i^2, p_i^2, e_i)_{i\in[\tau]}$ of $\mathcal{E}_{t_1}, \mathcal{E}_{t_2}$, respectively, we say that h_1, h_2 *can be combined into* $h = (X^i, \mathcal{C}_i, p_i, e_i)_{i\in[\tau]}$ if the following hold for every $i \in [\tau]$ (below, we denote by $d_i(u)$ the value $|\{uv \in E(X_t) \mid e_i(uv) = 1\}|$):

- $\mathcal{C}_i$ is the union of $\mathcal{C}_i^1$ and $\mathcal{C}_i^2$; and
- p_i can be obtained from p_i^1 and p_i^2 by analysing the degree d_i of the vertices in $W_i[X_t]$. More formally:

$$p_i(u) = \begin{cases} 0, & \text{if either } d_i(u) \text{ is odd and } \{p_i^1(u), p_i^2(u)\} = \{0, 1\}, \\ & \text{or } d_i(u) \text{ is even and } p_i^1(u) = p_i^2(u) \\ 1, & \text{if either } d_i(u) \text{ is odd and } p_i^1(u) = p_i^2(u), \\ & \text{or } d_i(u) \text{ is even and } \{p_i^1(u), p_i^2(u)\} = \{0, 1\}. \end{cases}$$

Lemma 17 *Consider a join node t with children t_1, t_2, and consider an entry $h = (X^i, \mathcal{C}_i, p_i, e_i)_{i\in[\tau]}$ of $\mathcal{E}_t$. Then, $\mathcal{E}_t(h) = 1$ if and only there exist entries h_1, h_2 of $\mathcal{E}_{t_1}, \mathcal{E}_{t_2}$, respectively, such that h_1, h_2 can be combined into h.*

Clearly, testing whether entries h_1, h_2 can be combined into h can be done in polynomial time. Again let $k = tw(G)$. Since there are at most $|X^i|^{|X^i|} = 2^{k\cdot\log k}$ partitions of X^i and at most 2^{k+1} vectors of size $|X_t|$ for each $i \in [\tau]$, and since we need to combine every pair of such entries in order to test whether they can be combined into h, we get a running time of $(2^{O(\tau\cdot k\cdot\log k)})^2 = 2^{O(\tau\cdot k\cdot\log k)}$ to compute entry $\mathcal{E}_t(h)$. One can see that this dominates the complexities for the other types of nodes. Therefore, because each table has size $(2^k \cdot 2^{k\log k} \cdot 2^k \cdot 2^{k^2})^\tau = 2^{O(k^2\cdot\tau)}$ and the tree decomposition has size $O(n)$, where n is equal to $|V(G)|$, our algorithm runs in time $2^{O(k^2\cdot\tau)} \cdot p(n)$, where p is a polynomial function.

As for Eulerian Trail and Eulerian Tour, it suffices to consider vectors e_i that do not allow for edge repetitions. Recall that our algorithm solves Partial Eulerian Local Trail to see that the obtained running time is as stated by Theorem 13.

7 Conclusions

In this paper we have investigated translations of Eulerian walks and trails in temporal terms. Eulerian walks traverse all the edges at least once, Eulerian trails traverse each edge exactly once, and Eulerian local trails traverse each edge at least once but never twice the same edge in the same snapshot. We have provided several NP-complete results, showing that all the corresponding decision problems are NP-complete even under very strict constraints. On the positive side, we have given an XP algorithm for Eulerian Walk when parameterized by τ, as well as an XP algorithm for Eulerian Trail and Eulerian Local Trail when parameterized by $\tau + tw(G)$. We ask

whether these complexities can be improved, i.e., whether Eulerian Walk can be solved in FPT time when parameterized by τ, and whether Eulerian Trail and Eulerian Local Trail can be solved in FPT time when parameterized by $\tau + tw(G)$. We recall that from our and previous results, the latter problems are para-NP-complete when parameterized either by τ or by $tw(G)$. Another possible question, and a broader one, is whether there are other possible interpretations of what an Eulerian temporal graph could be.

Additionally, as we have discussed previously, a natural generalization of the Eulerian trail problem on static graphs is the so called Chinese Postman problem. Since the latter is more general, it directly follows from the results presented here that translations in the temporal sense analogous to the ones given here for the Chinese Postman problem would already be NP-complete. However, it could be worth to investigate whether our polynomial cases would continue to be polynomial.

Finally, we mention that, looking at Table 1, one can find many open other problems either explicitly (e.g., what is the complexity of all the investigated problems on graphs with bounded treewidth when an edge is allowed to appear at most 3 times?), or related to the results presented (e.g., what other parameters would be promising for obtaining FPT algorithms?).

Acknowledgements A. Marino has been funded by MIUR under PRIN Project n. 20174LF3T8 AHeAD (Efficient Algorithms for HArnessing Networked Data), and by the University of Florence under Project GRANTED (GRaph Algorithms for Networked TEmporal Data). A. Silva has been funded by Grants CNPq/Brazil Produtividade no. 303803/2020-7, CNPq Universal 437841/2018-9 and CNPq/FUNCAP PRONEM no. PNE-0112-00061.01.00/16.

Funding Open access funding provided by Università degli Studi di Firenze within the CRUI-CARE Agreement.

Appendix A Quick Summary About Eulerian Tours in Dynamic Graphs

A *dynamic graph* is a pair (G, T) where G is a finite digraph and T is a function $T : E(G) \to \mathbb{Z}$, called *transit time function*. A dynamic graph can also be seen as a special type of infinite digraph $\mathcal{G}$, where $V(\mathcal{G}) = V(G) \times \mathbb{Z}$, and $(u, i)(v, j) \in E(\mathcal{G})$ if and only if $uv \in E(G)$ and $T(uv) = j - i$. Observe that the transit time of an arc can also be negative, and therefore there might exist arcs going from a vertex (u, i) to a vertex (v, j) with $j < i$, which in the temporal graph context would be considered as going back in time. An *Eulerian trail* in (G, T) is a trail that passes through all the edges of $\mathcal{G}$. More formally, it is a function $f : \mathbb{Z} \to V(G) \times \mathbb{Z}$ such

that $f(i)f(i+1) \in E(\mathcal{G})$ for every $i \in \mathbb{Z}$, and for every $(u,i)(v,j) \in E(\mathcal{G})$, there exists a unique ℓ such that $(u,i)(v,j)$ is equal to $f(\ell)f(\ell+1)$.

Recall that a digraph G is Eulerian if and only: (i) G has at most one non-trivial component; and (ii) the indegree of u, denoted with $d^-(u)$, is equal to its outdegree, denoted as $d^+(u)$, for every $u \in V(G)$. Observe that these conditions are also trivially necessary for the infinite case. Also, note that, given $u \in V(G)$, each out-arc uv of G leaving u gives rise to exactly one out-arc $(u,i)(v,i+T(u,v))$ leaving (u,i), for every $i \in \mathbb{Z}$. The same clearly holds for every in-arc. Therefore, one can see that $\mathcal{G}$ satisfies (i) and (ii) if and only if G satisfies (i) and (ii). However, as proved in [26], these are not the only necessary conditions. Nevertheless, a characterization is still possible, with an additional, also easy to test, condition.

Theorem 18 ([26]) *Let (G,T) be a dynamic graph. Then (G,T) has an Eulerian trail if and only the following conditions hold:*

1. $d^-(u) = d^+(u)$ *for every* $u \in V(G)$*;*
2. G *is connected; and*
3. $\sum_{e \in E(G)} T(e) \in \{-1, 1\}$

Even if the necessary part of the proof is more technical, the sufficiency part is quite natural, because condition (3) tells us that, given an Eulerian tour $T = (v_1, \ldots, v_m, v_1)$ of G and fixing a time i, we can use T to traverse all the edges incident to (v_1, i) in a way that we arrive in $(v_1, i+1)$ (or $(v_1, i-1)$ if the sum is -1) just in time to apply the same process to $(v_1, i+1)$. Because (v_1, i) is chosen arbitrarily, we are ensured to visit all edges of $\mathcal{G}$.

References

1. Akrida, E.C., Mertzios, G.B., Spirakis, P.G.: The temporal explorer who returns to the base. In: Heggernes, P. (ed.) Algorithms and Complexity—11th International Conference, CIAC 2019, Rome, Italy, May 27–29, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11485, pp. 13–24. Springer (2019)
2. Arumugam, S., Hamid, I., Abraham, V.: Decomposition of graphs into paths and cycles. J. Discrete Math. **2013**, 1–6 (2013)
3. Bodlaender, H.L., Drange, P.G., Dregi, M.S., et al.: A $c^k n$ 5-approximation algorithm for treewidth. SIAM J. Comput. **45**(2), 317–378 (2016)
4. Borgnat, P., Fleury, E., Guillaume, J., et al.: Evolving networks. In: Mining Massive Data Sets for Security, pp. 198–203 (2007)
5. Bumpus, B.M., Meeks, K.: Edge exploration of temporal graphs. In: Flocchini, P., Moura, L. (eds) Combinatorial Algorithms—32nd International Workshop, IWOCA 2021, Ottawa, ON, Canada, July 5–7, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12757, pp. 107–121. Springer (2021)
6. Casteigts, A., Flocchini, P., Quattrociocchi, W., et al.: Time-varying graphs and dynamic networks. Int. J. Parallel Emerg. Distrib. Syst. **27**(5), 387–408 (2012)
7. Çodur, M.K., Yılmaz, M.: A time-dependent hierarchical Chinese postman problem. CEJOR **28**(1), 337–366 (2020)
8. Cygan, M., Marx, D., Pilipczuk, M., et al.: Parameterized complexity of Eulerian deletion problems. Algorithmica **68**(1), 41–61 (2014)
9. Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., et al.: The complexity of multiterminal cuts. SIAM J. Comput. **23**(4), 864–894 (1994)
10. Dror, M., Stern, H., Trudeau, P.: Postman tour on a graph with precedence relation on arcs. Networks **17**(3), 283–294 (1987)

11. Erlebach, T., Spooner, J.T.: Faster exploration of degree-bounded temporal graphs. In: Potapov I, Spirakis PG, Worrell J (eds) 43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27–31, 2018, Liverpool, UK, LIPIcs, vol 117. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 36:1–36:13 (2018)
12. Erlebach, T., Spooner, J.T.: Non-strict temporal exploration. In: Richa, A.W., Scheideler, C. (eds) Structural Information and Communication Complexity—27th International Colloquium, SIROCCO 2020, Paderborn, Germany, June 29–July 1, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12156, pp. 129–145. Springer (2020)
13. Erlebach, T., Hoffmann, M., Kammer, F.: On temporal graph exploration. In: 42nd International Colloquium on Automata, Languages, and Programming—ICALP 2015, Kyoto, Japan, Lecture Notes in Computer Science, vol. 9134, pp. 444–455. Springer (2015)
14. Euler, L.: Solutio problematis ad geometriam situs pertinentis. Commentarii academiae scientiarum Petropolitanae pp. 128–140 (1741)
15. Fomin, F.V., Golovach, P.A.: Long circuits and large Euler subgraphs. SIAM J. Discret. Math. **28**(2), 878–892 (2014)
16. Garey, M.R., Johnson, D.S., Tarjan, R.E.: The planar Hamiltonian circuit problem is NP-complete. SIAM J. Comput. **5**(4), 704–714 (1976)
17. Gómez, R., Wakabayashi, Y.: Covering a graph with nontrivial vertex-disjoint paths: Existence and optimization. In: Brandstädt, A., Köhler, E., Meer, K. (eds) Graph-Theoretic Concepts in Computer Science—44th International Workshop, WG 2018, Cottbus, Germany, June 27–29, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11159, pp. 228–238. Springer (2018)
18. Guan, M.: Graphic programming using odd or even points. Acta Mathematica Sinica (in Chinese) 10:263–266. Translated in Chinese Mathematics 1. Am. Math. Soc. **1962**, 273–277 (1960)
19. Kempe, D., Kleinberg, J.M., Kumar, A.: Connectivity and inference problems for temporal networks. In: Yao FF, Luks EM (eds) Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21–23, pp. 504–513 . ACM, Portland (2000)
20. Kloks, T.: Treewidth, Computations and Approximations. Lecture Notes in Computer Science, vol. 842. Springer (1994)
21. Latapy, M., Viard, T., Magnien, C.: Stream graphs and link streams for the modeling of interactions over time. Soc. Netw. Anal. Min. **8**(1), 61:1-61:29 (2018)
22. Manuel, P.: Revisiting path-type covering and partitioning problems. arXiv:1807.10613 (2018)
23. Marino, A., Silva, A.: Königsberg sightseeing: Eulerian walks in temporal graphs. In: Flocchini, P., Moura, L. (eds) Combinatorial Algorithms - 32nd International Workshop, IWOCA 2021, Ottawa, ON, Canada, July 5–7, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12757, pp. 485–500. Springer (2021)
24. Michail, O.: An introduction to temporal graphs: an algorithmic perspective. Internet Math. **12**(4), 239–280 (2016)
25. Michail, O., Spirakis, P.G.: Traveling salesman problems in temporal graphs. Theoret. Comput. Sci. **634**, 1–23 (2016)
26. Orlin, J.B.: Some problems on dynamic/periodic graphs. In: Progress in Combinatorial Optimization, pp. 273–293. Elsevier (1984)
27. Sun, J., Tan, G., Qu, H.: Dynamic programming algorithm for the time dependent Chinese postman problem. J. Inf. Comput. Sci. **8**, 833–841 (2011)
28. Wang, H.F., Wen, Y.P.: Time-constrained Chinese postman problems. Comput. Math. Appl. **44**(3–4), 375–387 (2002)
29. West, D.: Introduction to Graph Theory, vol. 2. Prentice Hall, Upper Saddle River (1996)