# Counting Temporal Paths

**Jessica Enright**
School of Computing Science, University of Glasgow, UK

**Kitty Meeks**
School of Computing Science, University of Glasgow, UK

**Hendrik Molter**
Department of Computer Science and Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer-Sheva, Israel

## Abstract

The betweenness centrality of a vertex $v$ is an important centrality measure that quantifies how many optimal paths between pairs of other vertices visit $v$. Computing betweenness centrality in a temporal graph, in which the edge set may change over discrete timesteps, requires us to count temporal paths that are optimal with respect to some criterion. For several natural notions of optimality, including *foremost* or *fastest* temporal paths, this counting problem reduces to #Temporal Path, the problem of counting *all* temporal paths between a fixed pair of vertices; like the problems of counting foremost and fastest temporal paths, #Temporal Path is #P-hard in general. Motivated by the many applications of this intractable problem, we initiate a systematic study of the parameterised and approximation complexity of #Temporal Path. We show that the problem presumably does not admit an FPT-algorithm for the feedback vertex number of the static underlying graph, and that it is hard to approximate in general. On the positive side, we prove several exact and approximate FPT-algorithms for special cases.

**2012 ACM Subject Classification** Theory of computation → Graph algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Approximation algorithms analysis; Mathematics of computing → Discrete mathematics

**Keywords and phrases** Temporal Paths, Temporal Graphs, Parameterised Counting, Approximate Counting, #P-hard Counting Problems, Temporal Betweenness Centrality

**Digital Object Identifier** 10.4230/LIPIcs.STACS.2023.30

**Related Version** *Full Version*: `https://arxiv.org/abs/2202.12055`

**Funding** *Jessica Enright*: Supported by EPSRC grant EP/T004878/1.
*Kitty Meeks*: Supported by EPSRC grants EP/T004878/1 and EP/V032305/1.
*Hendrik Molter*: Supported by the ISF, grants No. 1456/18 and No. 1070/20, and European Research Council, grant number 949707.

**Acknowledgements** This work was initiated at the Dagstuhl Seminar "Temporal Graphs: Structure, Algorithms, Applications" (Dagstuhl Seminar Nr. 21171).

## 1 Introduction

Computing a (shortest) path between two vertices in a graph is one of the most important tasks in algorithmic graph theory and serves as a subroutine in a wide variety of algorithms for connectivity-related graph problems. The *betweenness centrality* measure for vertices in a graph was introduced by Freeman [23] and motivates the task of *counting* shortest paths in a graph. Intuitively, betweenness centrality measures the importance of a vertex for information flow under the assumption that information travels along optimal (i.e. shortest) paths. More formally, the betweenness of a vertex $v$ is based on the ratio of the number of shortest paths between vertex pairs that visit $v$ as an intermediate vertex and the total number of shortest paths, thus its computation is closely related to shortest path counting.



The betweenness centrality is a commonly used tool in network analysis and it can be computed in polynomial time; e.g. Brandes' algorithm [9] serves as a blueprint for all modern betweenness computation algorithms and implicitly also counts shortest paths.

In contrast to the tractability of counting shortest paths, the problem of counting *all* paths between two vertices in a graph is one of the classic problems discussed in the seminal paper by Valiant [46] that is complete for the complexity class #P (the counting analogue of NP) and hence is presumably not doable in polynomial time.

*Temporal* graphs are a natural generalisation of graphs that capture dynamic changes over time in the edge set. They have a fixed vertex set and a set of *time-edges* which have integer time labels indicating at which time(s) they are active. In recent years, the research field of studying algorithmic problems on temporal graphs has steadily grown [28, 29, 34, 35]. In particular, an additional layer of complexity is added to connectivity related problems in the temporal setting. Paths in temporal graphs have to respect time, that is, a *temporal path* has to traverse time-edges with non-decreasing time labels [32][1]. This implies that temporal connectivity is generally not symmetric and not transitive, a major difference from the non-temporal case. Furthermore, there are several natural optimality concepts for temporal paths, the most important being *shortest*, *foremost*, and *fastest* temporal paths [10]. Intuitively speaking, shortest temporal paths use a minimum number of time-edges, foremost temporal paths arrive as early as possible, and fastest temporal paths have a minimum difference between start and arrival times. We remark that an optimal path with respect to any of these three criteria can be found in polynomial time [10, 47]. The existence of multiple natural optimality concepts for temporal paths implies several natural definitions of temporal betweenness, one for each path optimality concept [40, 12, 34].

Similar to the non-temporal case, the ability to *count* optimal temporal paths is a key ingredient for the corresponding temporal betweenness computation. However, the picture is more complex in the temporal setting. Shortest temporal paths can be counted in polynomial time and the corresponding temporal betweenness can be computed in polynomial time [12, 33, 27, 40]. In contrast, counting foremost or fastest temporal paths is #P-hard [39, 12, 36], which implies that computing the corresponding temporal betweenness is #P-hard as well [12]. Indeed, Buß et al. [12] show that there is a polynomial time reduction from the problem of counting foremost or fastest temporal paths to the problem of the corresponding temporal betweenness computation. Note that a reduction in the other direction is straightforward.

In this work, we study the (parameterised) computational complexity of (approximately) counting foremost or fastest temporal paths. In fact, we study the simpler and arguably more natural problem of counting all temporal paths from a start vertex $s$ to a destination vertex $z$ in a temporal graph.

Let $\mathcal{G} = (V, \mathcal{E}, T)$ denote a temporal graph with vertex set $V$, time-edge set $\mathcal{E}$, and maximum time label (or lifetime) $T$ (formal definitions are given in Section 2). We are then concerned with the following computational problem:

| #Temporal Path | |
|---|---|
| Input: | A temporal graph $\mathcal{G} = (V, \mathcal{E}, T)$ and two vertices $s, z \in V$. |
| Task: | Count the temporal $(s, z)$-paths in $\mathcal{G}$. |

[1] Temporal paths that traverse time-edge with non-decreasing time labels are often referred to as "non-strict", in contrast to *strict* temporal paths, which traverse time-edges with increasing time labels. In this work, we focus on non-strict temporal paths.

It is easy to see that #Temporal Path generalises the problem of counting paths in a non-temporal graph (all time-edges have the same time label), hence we deduce that #Temporal Path is #P-hard. Furthermore, observe that using an algorithm for #Temporal Path, it is possible to count foremost or fastest temporal paths with only polynomial overhead in the running time; we discuss this reduction in more detail in Section 2.3. Hence, all exact algorithms we develop for #Temporal Path can be used to compute the temporal betweenness based on foremost or fastest temporal paths with polynomial overhead in the running time. To the best of our knowledge, this is the first attempt to systematically study the parameterised complexity and approximability of #Temporal Path.

## 1.1 Related Work

As discussed above, the temporal setting adds a new dimension to connectivity-related problems. The problems of computing shortest, foremost, and fastest temporal paths have been studied thoroughly [10, 47, 5]. The temporal setting also offers room for new natural temporal path variants that do not have an analogue in the non-temporal setting. Casteigts et al. [13] study the problem of finding *restless temporal paths* that dwell an upper-bounded number of time steps in each vertex, while Füchsle et al. [24] study the problem of finding *delay-robust routes* in a temporal graph (intuitively, temporal paths that are robust with respect to edge delays); both problems turn out to be NP-hard.

The problem of *counting* (optimal) temporal paths has mostly been studied indirectly in the context of temporal betweenness computation. However, the computation of temporal betweenness has received much attention [12, 39, 40, 33, 44, 27, 43, 2, 45, 38, 41]. Most of the mentioned work considers temporal betweenness variants that are polynomial-time computable. The corresponding optimal temporal paths are mostly shortest temporal paths or variations thereof. There are at least three notable exceptions: Buß et al. [12] also consider *prefix-foremost* temporal paths and the corresponding temporal betweenness and show that the latter is computable in polynomial time. Furthermore they show #P-hardness for several temporal betweenness variants based on *strict* optimal temporal paths. Rad et al. [39] consider temporal betweenness based on foremost temporal paths and show that its computation is #P-hard. They further give an FPT-algorithm to compute temporal betweenness based on foremost temporal paths for the number of vertices as a parameter (note that the size of a temporal graph generally cannot be bounded by a function of the number of its vertices). Rymar et al. [40] give a quite general sufficient condition called *prefix-compatibility* for optimality concepts for temporal paths that makes it possible to compute the corresponding temporal betweenness in polynomial time.

In the static setting the general problem of counting $(s, z)$-paths in static graphs is known to be #P-complete [46]. In the parameterised setting, the problem of counting length-$k$ paths (with parameter $k$) was one of the first problems shown to be #W[1]-complete [21], but the problem does admit an efficient parameterised approximation algorithm [3]. It is also generally considered folklore that the problem of counting paths (of any length) admits an FPT-algorithm parameterised by the treewidth of the input graph.

## 1.2 Our Contribution

Our goal is to initiate the systematic study of the parameterised and approximation complexity of #Temporal Path. We provide an argument that #Temporal Path is essentially equivalent to counting foremost or fastest paths or computing the respective temporal betweenness centrality in Section 2.3. Due to space constraints, proofs of results marked with ($\star$) are (partially) deferred to the full version of this work [19].

**Hardness results (Section 3).** The main technical contribution of this paper is a reduction showing that #Temporal Path is intractable even when very strong restrictions are placed on the underlying graph; specifically the problem is hard for $\oplus$W[1] when parameterised by the feedback vertex number of the underlying graph, which rules out the existence of FPT algorithms with respect to several common parameters. We also show that it is NP-hard even to approximate the number of temporal $(s, z)$-paths in general, motivating the study of approximate counting in more restricted settings.

**Exact algorithms for special cases (Section 4).** We show that the problem is polynomial-time solvable if the underlying graph is a forest, and then use a wide range of algorithmic techniques to generalise this result in different ways. We show that the problem is fixed-parameter tractable with respect to two "distance to forest" parameterisations that are larger than the feedback vertex number of the underlying graph (timed feedback vertex number and underlying feedback edge number). We further show that #Temporal Path is in FPT parameterised by the treewidth of the underlying graph and the lifetime combined, or parameterised by the recently introduced parameter "vertex-interval-membership-width".

**Approximation algorithms (Section 5).** We show that there is an FPTRAS for #Temporal Path parameterised by the maximum permitted length of a temporal $(s, z)$-path. We then turn our attention to the problem of approximating betweenness centrality, as the relationship between path counting and computing betweenness is not so straightforward in the approximate setting: we demonstrate that, whenever there exists an FPRAS (respectively FPTRAS) for #Temporal Path, we can efficiently approximate the maximum betweenness centrality of any vertex in the temporal graph. These two results together give an FPTRAS to estimate the maximum betweenness centrality of any vertex in a temporal graph (with respect to either foremost or fastest temporal paths) parameterised by the vertex cover number or treedepth of the underlying input graph.

## 2 Preliminaries and Basic Observations

In this section we provide all basic notations, definitions, and terminology used in this work. We discuss the relation between temporal path counting and temporal betweenness computation in more detail in Section 2.3. We use standard definitions and terminology from parameterised complexity theory [18, 22, 16, 3, 21]. Additional background on parameterised and approximate counting complexity are given in the full version of this work [19].

Given a static graph $G = (V, E)$, we say that a sequence $P = (\{v_{i-1}, v_i\})_{i=1}^{k}$ of edges in $E$ forms a *path* in $G$ if $v_i \neq v_j$ for all $0 \leq i < j \leq k$.

### 2.1 Temporal Graphs and Paths

There are several different definitions and notations used in the context of temporal graphs [28, 29, 34, 35] which are mostly equivalent. Here, we use the following definitions and notations:

An (undirected, simple) *temporal graph* with lifetime $T \in \mathbb{N}$ is a tuple $\mathcal{G} = (V, \mathcal{E}, T)$, with time-edge set $\mathcal{E} \subseteq \binom{V}{2} \times [T]$. We assume all temporal graphs in this paper to be undirected and simple. The *underlying graph* of $\mathcal{G}$ is defined as the static graph $G = (V, \{\{u, v\} \mid \exists t \in [T] \text{ s.t. } (\{u, v\}, t) \in \mathcal{E}\})$. We denote by $E_t$ the set of edges of $G$ that are active at time $t$, that is, $E_t = \{\{u, v\} \mid (\{u, v\}, t) \in \mathcal{E}\}$.

For every $v \in V$ and every time step $t \in [T]$, we denote the *appearance of vertex $v$ at time $t$* by the pair $(v,t)$. For a time-edge $(\{v,w\},t)$ we call the vertex appearances $(v,t)$ and $(w,t)$ its *endpoints* and we call $\{v,w\}$ its *underlying edge*.

We assume that every number in $[T]$ appears at least once as a label for an edge in $\mathcal{E}$. In other words, we ignore labels that are not used for any edges since they are irrelevant for the problems we consider in this work. It follows that we assume $T \leq |\mathcal{E}|$ and hence $T \in \mathcal{O}(|\mathcal{G}|) = \mathcal{O}(|V| + |\mathcal{E}|)$.

A *temporal $(s,z)$-path* (or *temporal path*) of length $k$ from vertex $s = v_0$ to vertex $z = v_k$ in a temporal graph $\mathcal{G} = (V, \mathcal{E}, T)$ is a sequence $P = ((\{v_{i-1}, v_i\}, t_i))_{i=1}^{k}$ of time-edges in $\mathcal{E}$ such that the corresponding sequence of underlying edges forms a path in the underlying graph of $\mathcal{G}$ and, for all $i \in [k-1]$, we have that $t_i \leq t_{i+1}$. Given a temporal path $P = ((\{v_{i-1}, v_i\}, t_i))_{i=1}^{k}$, we denote the set of vertices of $P$ by $V(P) = \{v_0, v_1, \ldots, v_k\}$ and we say that $P$ *visits* the vertex $v_i$ if $v_i \in V(P)$. Moreover, we call vertex appearances $(v_{i-1}, t_i)$ *outgoing* for $P$ and we call the vertex appearances $(v_i, t_i)$ *incoming* for $P$. Note that, if $t_i = t_{i+1}$, then $(v_i, t_i)$ is both incoming and outgoing for $P$. We define $(v_0, 1)$ to be incoming for $P$ and $(v_k, T)$ to be outgoing for $P$. We say that a vertex appearance is *visited* by $P$ if it is outgoing or incoming for $P$ (so a vertex is visited by $P$ if and only if at least one of its appearances is visited by $P$). We say that $P$ *starts* at $v_0$ at time $t_1$ and *arrives* at $v_k$ at time $t_k$. We say that $P'$ is a *temporal subpath* of $P$ if $P'$ is a subsequence of $P$. Furthermore, we define the following optimality concepts for temporal $(s,z)$-paths $P$.

- $P$ is a *shortest* temporal $(s,z)$-path if there is no temporal path $P'$ from $s$ to $z$ such that the length of $P'$ is strictly less than the length of $P$.
- $P$ is a *foremost* temporal $(s,z)$-path if there is no temporal path $P'$ from $s$ to $z$ such that $P'$ arrives at $z$ at a strictly smaller time than $P$.
- $P$ is a *fastest* temporal $(s,z)$-path if there is no temporal path $P'$ from $s$ to $z$ such that the difference between the time at which $P'$ starts at $s$ and the time at which $P'$ arrives at $z$ is strictly smaller than the analogous difference of times for $P$.

## 2.2 Temporal Betweenness Centrality

We follow the notation and definition for temporal betweenness given by Buß et al. [12]. Let $\mathcal{G} = (V, \mathcal{E}, T)$ be a temporal graph. For any $s, z \in V$, $\sigma_{sz}^{(\star)}$ is the number of $\star$-optimal temporal paths from $s$ to $z$. We define $\sigma_{vv}^{(\star)} := 1$. For any vertex $v \in V$, we write $\sigma_{sz}^{(\star)}(v)$ for the number of $\star$-optimal paths that pass through $v$. We set $\sigma_{sz}^{(\star)}(s) := \sigma_{sz}^{(\star)}$ and $\sigma_{sz}^{(\star)}(z) := \sigma_{sz}^{(\star)}$.

We do not assume that there is a temporal path from any vertex to any other vertex in the graph. To determine between which (ordered) pairs of vertices a temporal path exists, we use a *connectivity matrix* $A$ of the temporal graph: let $A$ be a $|V| \times |V|$ matrix, where for every $v, w \in V$ we have that $A_{v,w} = 1$ if there is a temporal path from $v$ to $w$, and $A_{v,w} = 0$ otherwise. Note that $A_{s,z} = 1$ if and only if $\sigma_{sz}^{(\star)} \neq 0$. Formally, temporal betweenness based on $\star$-optimal temporal paths is defined as follows.

▶ **Definition 1** (Temporal Betweenness). *The* temporal betweenness *of any vertex $v \in V$ is given by:*

$$C_B^{(\star)}(v) := \sum_{s \neq v \neq z \text{ and } A_{s,z}=1} \frac{\sigma_{sz}^{(\star)}(v)}{\sigma_{sz}^{(\star)}}.$$

## 2.3 Temporal Betweenness vs. Temporal Path Counting

In this subsection we discuss the relationship between the problems of computing temporal betweenness and counting temporal paths. We show that we can compute temporal betweenness based on foremost and fastest temporal paths using an algorithm for #Temporal Path with only polynomial overhead in the running time. Let $\mathcal{G} = (V, \mathcal{E}, T)$ be a temporal graph. We start with the following easy observation.

▶ **Observation 2.** *Given an algorithm to count all $\star$-optimal temporal $(s, z)$-paths in $\mathcal{G}$ in time $t(\mathcal{G})$, we can compute the temporal betweenness based on $\star$-optimal temporal paths of any vertex of $\mathcal{G}$ in $t(\mathcal{G}) \cdot |\mathcal{G}|^{\mathcal{O}(1)}$ time.*

This follows by observing that we can count the number of temporal $(s, z)$-paths in $\mathcal{G}$ that visit a vertex $v$ by first counting all temporal $(s, z)$-paths in $\mathcal{G}$ and then subtracting the number of temporal $(s, z)$-paths in $\mathcal{G} - \{v\}$.

Next we observe that we can count foremost and fastest temporal $(s, z)$-paths using an algorithm for #Temporal Path, with only polynomial overhead.

▶ **Observation 3.** *Given an algorithm for #Temporal Path that runs in time $t(\mathcal{G})$, we can compute all foremost temporal $(s, z)$-paths and all fastest temporal $(s, z)$-paths in $t(\mathcal{G}) \cdot |\mathcal{G}|^{\mathcal{O}(1)}$ time.*

First, note that we can compute a foremost temporal $(s, z)$-path and a fastest temporal $(s, z)$-path in polynomial time [10, 47]. In the case of foremost temporal $(s, z)$-paths, we can in this way obtain the time at which a foremost temporal $(s, z)$-path arrives at $z$ and remove all time-edges with later time labels from $\mathcal{G}$. After this modification, every temporal $(s, z)$-path is foremost hence we can count them using an algorithm for #Temporal Path.

In the case of fastest temporal $(s, z)$-paths, we can in the same way obtain the time difference $t_f$ between starting at $s$ and arriving at $z$ for any fastest temporal $(s, z)$-path. We can now iterate over all intervals $[t_0, t_0 + t_f]$ with $1 \leq t_0 \leq T - t_f$ and, for each one, create an instance of #Temporal Path by removing all time-edges from $\mathcal{G}$ that are either earlier than $t_0$ or later than $t_0 + t_f$. After this modification, every temporal $(s, z)$-path in the instance corresponding to any interval is fastest, and every fastest temporal path survives in exactly one instance; hence we can count fastest temporal paths by calling an algorithm for #Temporal Path on each instance and summing the results.

Using Observations 2 and 3 we obtain the following lemma, which implies that our polynomial-time and FPT-algorithms for special cases of #Temporal Path yield polynomial-time solvability and fixed-parameter tractability results respectively for temporal betweenness based on foremost temporal paths or fastest temporal paths, under the same restrictions.

▶ **Lemma 4.** *Given an algorithm for #Temporal Path that runs in time $t(\mathcal{G})$, we can can compute the temporal betweenness based on foremost temporal paths or fastest temporal paths of any vertex of $\mathcal{G}$ in $t(\mathcal{G}) \cdot |\mathcal{G}|^{\mathcal{O}(1)}$ time.*

If we can only count temporal paths *approximately*, however, the relationship between temporal path counting and temporal betweenness computation is not so straightforward. In the exact setting, we were able to determine the number of temporal $(s, z)$-paths visiting $v$ by calculating the difference between the number of temporal $(s, z)$-paths in $\mathcal{G}$ and $\mathcal{G} - \{v\}$ respectively. However, in the approximate setting, we cannot use the same strategy: if there are $N$ temporal paths in total and $N_{-v}$ is an $\varepsilon$-approximation to the number of temporal paths that do not contain $v$, it does not follow that $N - N_{-v}$ is an $\varepsilon$-approximation to the number of temporal paths that do contain $v$, as the relative error will potentially be much

higher if the proportion of temporal paths containing $v$ is very small. A similar issue arises if we aim to estimate the number of temporal paths through $v$ by sampling a collection of temporal paths (from an approximately uniform distribution) and using the proportion that contain $v$ as an estimate for the total proportion of temporal paths containing $v$: if the proportion that contain $v$ is exponentially small, we would need exponentially many samples to have a non-trivial probability of finding at least one temporal path which does contain $v$; otherwise we deduce incorrectly that there are no temporal paths through $v$ and output 0, which cannot be an $\varepsilon$-approximation of a non-zero number of temporal paths for any $\varepsilon < 1$.

Lastly, we briefly shift our attention to computational hardness. Buß et al. [12] provide a reduction from #Temporal Path to the computation of temporal betweenness based on foremost temporal paths and to the computation of temporal betweenness based on fastest temporal paths. In both cases, three new vertices are added to the temporal graph and all newly added time-edges are incident with at least one of the newly added vertices. This implies that our parameterised hardness result in the next section (Theorem 5) also holds for temporal betweenness computation based on foremost temporal paths or fastest temporal paths, since the reductions by Buß et al. [12] increase the feedback vertex number of the underlying graph by at most three.

## 3 Intractability Results for Temporal Path Counting

In this section we prove two hardness results for #Temporal Path. In Section 3.1 we demonstrate parameterised intractability with respect to the feedback vertex number of the underlying graph. We follow this in Section 3.2 with an easy reduction demonstrating that the classical #P-complete #Path problem [46] (definition as below) is unlikely to admit an FPRAS in general, which straightforwardly implies the same result for #Temporal Path.

| #Path | |
|---|---|
| Input: | A graph $G = (V, E)$ and two vertices $s, z \in V$. |
| Task: | Compute the number of paths from $s$ to $z$ in $G$. |

### 3.1 Parameterised Hardness

In this section we present our main parameterised hardness result, which provides strong evidence that #Temporal Path does not admit an FPT algorithm when parameterised by the feedback vertex number of the underlying graph. Note that this also rules out FPT algorithms for many other parameterizations, including the treewidth of the underlying graph. However, it is folklore that #Path admits an FPT algorithm parameterised by treewidth as a parameter (this is also implied by our result Theorem 14). The result here, therefore, means that #Temporal Path is strictly harder than #Path in terms of parameterised complexity for the parameterisations that are at most the feedback vertex number of the underlying graph and at least the treewidth of the underlying graph.

▶ **Theorem 5** (⋆). *#Temporal Path is $\oplus$W[1]-hard when parameterised by the feedback vertex number of the underlying graph.*

**Proof.** We present a parameterised counting Turing reduction from $\oplus$Multicoloured Independent Set on 2-Track Interval Graphs parameterised by the number of colours $k$. In $\oplus$Multicoloured Independent Set on 2-Track Interval Graphs we are given a set $I$ of interval pairs and a colouring function $c : I \to [k]$ and asked whether there is an odd number of $k$-sized sets of interval pairs in $I$ such that in each set,

every two interval pairs have different colours and are non-intersecting. Two interval pairs $([x_a, x_b], [x_{a'}, x_{b'}]), ([y_a, y_b], [y_{a'}, y_{b'}])$ are considered non-intersecting if $[x_a, x_b] \cap [y_a, y_b] = \emptyset$ and $[x_{a'}, x_{b'}] \cap [y_{a'}, y_{b'}] = \emptyset$.

Inspecting the W[1]-hardness proof by Jiang [31] for Independent Set on 2-Track Interval Graphs shows that the reduction used from Multicoloured Clique parameterised by the number of colours $k$ is parsimonious[2] and the reduction also shows W[1]-hardness for the multicoloured version of the problem. Since $\oplus$Multicoloured Clique is $\oplus$W[1]-hard when parameterised by the number of colours $k$ [8], we can conclude that $\oplus$Multicoloured Independent Set on 2-Track Interval Graphs is $\oplus$W[1]-hard when parameterised by the number of colours $k$.

Given an instance $(I, c)$ of $\oplus$Multicoloured Independent Set on 2-Track Interval Graphs, where $I$ is a set of interval pairs and $c : I \to [k]$ is a colouring function, we create $\mathcal{O}(2^k)$ temporal graphs. We assume w.l.o.g. that for all $([x_a, x_b], [x_{a'}, x_{b'}]), ([y_a, y_b], [y_{a'}, y_{b'}]) \in I$ that $|\{x_a, x_b, y_a, y_b\}| = 4$ and $|\{x_{a'}, x_{b'}, y_{a'}, y_{b'}\}| = 4$ or $([x_a, x_b], [x_{a'}, x_{b'}]) = ([y_a, y_b], [y_{a'}, y_{b'}])$, that is, if two interval pairs are different, we assume that all endpoints on each track are pairwise different. Furthermore, we assume w.l.o.g. that all intervals contained in pairs in $I$ are integer subsets of $[2|I|]$. The main intuition of our construction follows:

- We model track one with a path in the underlying graph and track two with time.
- Through the feedback vertices of the underlying graph, a temporal path can "enter" and "leave" the path that models track one.
- The number of feedback vertices corresponds to the number of colours.
- We have to make sure that we can determine the parity of the number of temporal paths visiting all feedback vertices.
- The number of temporal paths that do not correspond to independent sets should not be considered. It seems difficult to get an exact handle on the number of such paths, however we will show that this number is even. Note that, intuitively, this is the main reason we show hardness for $\oplus$W[1] and not #W[1].

We construct a family of *directed* temporal graphs $(\mathcal{G}^{\mathcal{C}} = (V, \mathcal{A}^{\mathcal{C}}, 2|I| + 1))_{\mathcal{C} \subseteq [k]}$ with rational time labels (such that the maximum time label is at most $2|I| + 1$), where $\mathcal{A}^{\mathcal{C}} \subseteq V \times V \times \mathbb{Q}$ for all $\mathcal{C} \subseteq [k]$. Towards the end of the proof we explain how to remove the need for directed edges which will also have the consequence that the temporal graphs only contain strict temporal paths. Note that we can scale up the lifetime to remove the need for rational time labels, however using rational time labels will be convenient in the construction and the correctness proof.

- We set $V := V_I \cup \{s, z', z\} \cup \{w_1, \ldots, w_k\} \cup \{u_x \mid x \in I\}$, where $V_I := \{v_1, \ldots, v_{2|I|}\}$.
- We set $\mathcal{A}^{\mathcal{C}} := \bigcup_{x \in I \wedge c(x) \in \mathcal{C}} \mathcal{A}_x \cup \{(s, w_i, 1) \mid i \in [k]\} \cup \{(z', z, 2|I| + 1)\}$, where

$$\begin{aligned}\mathcal{A}_x :=& \{(w_{c(x)}, u_x, a'), (u_x, v_a, b'), (u_x, v_a, b' + 1 - a\varepsilon), (v_b, z', b')\} \\ &\cup \{(v_b, w_i, b') \mid i \in [k] \wedge i \neq c(x)\} \\ &\cup \{(v_j, v_{j+1}, b'), (v_j, v_{j+1}, b' + 1 - (j+1)\varepsilon) \mid j \in \{a, \ldots, b-1\}\}\end{aligned}$$

for $x = ([a, b], [a', b']) \in I$ and $\varepsilon = \frac{1}{2|I|}$.

[2] Informally speaking, parsimonious reductions do not change the number of solutions.

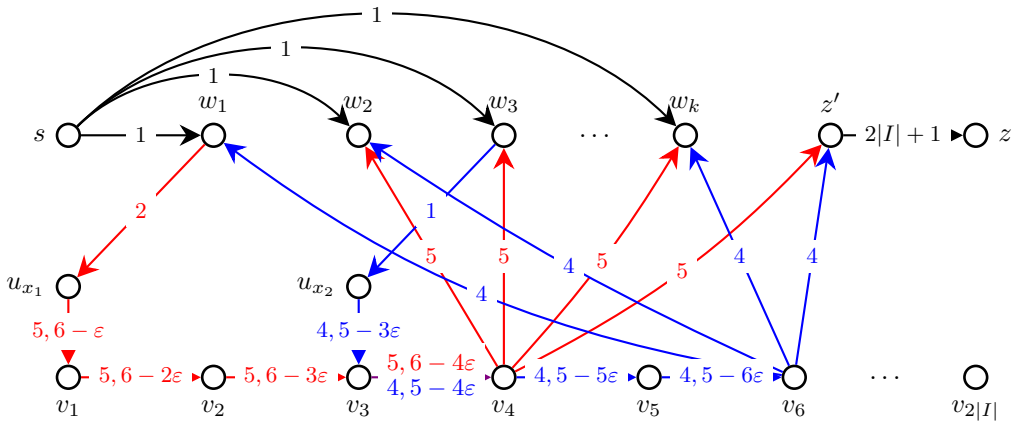


**Figure 1** Illustration of $\mathcal{G}^{\mathcal{C}}$ with $\mathcal{C} = \{1, 3\}$ and two interval pairs $x_1, x_2 \in I$ where $x_1 = ([1, 4], [2, 5])$ and $x_2 = ([3, 6], [1, 4])$, and the corresponding colours are $c(x_1) = 1$ and $c(x_2) = 3$. The arcs added for $x_1$ are depicted in red and the arcs added for $x_2$ are depicted in blue.

For all $\mathcal{G}^{\mathcal{C}}$ we use $s$ as the starting vertex and $z$ as the end vertex of the temporal paths we want to count. The temporal graphs $\mathcal{G}^{\mathcal{C}}$ can each clearly be constructed in polynomial time and it is easy to see that the vertex set $\{s, z', z, w_1, \ldots, w_k\}$ constitutes a feedback vertex set of size $\mathcal{O}(k)$ for each of them (even if edge directions are removed). The construction is illustrated in Figure 1. The correctness proof of the reduction is deferred to the full version of this work [19]. ◀

## 3.2 Approximation Hardness

In this section, we prove the following result.

▶ **Theorem 6.** *There is no fully polynomial randomised approximation scheme (FPRAS) for #Temporal Path unless randomised polynomial time (RP) equals NP.*

It is straightforward to reduce from #Path, the problem of counting $(s, z)$-paths in a static graph, to #Temporal Path: we set every edge to be active at time one only. Hardness of #Path is proved easily by imitating the reduction used by Jerrum et al. [30] to demonstrate that there is no FPRAS to count directed cycles in a directed graph.[3] We note that the reduction also rules out the existence of any polynomial-time (randomised) approximation algorithm achieving any polynomial additive error.

▶ **Theorem 7** (⋆). *There is no FPRAS for #Path unless RP=NP.*

# 4 Exact Algorithms for Temporal Path Counting

In this section, we present several exact algorithms for #Temporal Path. We start in Section 4.1 with a polynomial-time algorithm for temporal graphs that have a forest as underlying graph. In Section 4.2 we show that our polynomial-time algorithm can be generalised in two ways, obtaining FPT-algorithms for the so-called timed feedback vertex

[3] Indeed, the fact that this technique can be adapted to demonstrate the hardness of approximately counting $(s, z)$-paths is noted without proof by Sinclair [42].

number and the feedback edge number of the underlying graph. In Section 4.3 we show that #Temporal Path is in FPT when parameterised by the treewidth of the underlying graph and the lifetime combined. Lastly, in Section 4.4, we give an FPT algorithm for #Temporal Path parameterised by the so-called vertex-interval-membership-width.

## 4.1 A Polynomial Time Algorithms for Forests

As a warm-up, we note that #Temporal Path can be solved in polynomial time with a simple dynamic program if the underlying graph is a forest. This is used as a subroutine for algorithms presented in Section 4.2.

▶ **Theorem 8.** *#Temporal Path is solvable in $\mathcal{O}(|V| \cdot T^2)$ time if the underlying graph of the input temporal graph is a forest.*

**Proof.** Let $\mathcal{G} = (V, \mathcal{E}, T)$ together with two verices $s, z \in V$ be an an instance of #Temporal Path. We argue that this instance can be solved in polynomial time if there is a unique path between $s$ and $z$ in the underlying graph of $\mathcal{G}$. Note that this is the case if the underlying graph of $\mathcal{G}$ is a forest.

First, observe that when counting $(s, z)$-paths starting at $s$ and arriving at $z$, if there is a unique static path between $s$ and $z$ in the underlying graph then we need only consider time-edges between vertices of that unique static path in our temporal graph when counting, as our temporal path may not repeat vertices and so corresponds to a path in the underlying graph. Edges not lying on the unique static path between $s$ and $z$ can therefore be deleted without changing the result, so we may w.l.o.g. consider an instance in which the underlying graph consists only of a static path $P = (v_0, v_1, ..., v_{|P|})$ with $s = v_0$ and $z = v_{|P|}$ as the leaf vertices.

We will base our counting on a recording at each vertex $v_i$ in $P$ of how many temporal $(s, v_i)$-paths there are starting at $s$ and arriving at $v_i$ at time $t$ or earlier. Note that there are $\mathcal{O}(|P| \cdot T) = \mathcal{O}(|V| \cdot T)$ such vertex-time pairs.

We argue by induction on $i$ that we can correctly compute this number for every vertex-time pair by dynamic programming. As a base case, note that there is one path from $s$ to $s$ for any arrival time. Then we assume that we have these numbers computed correctly for some $v_i$ with $i \geq 0$ and show how we compute them for $v_{i+1}$. Formally, our dynamic program is defined as follows.

$$F(v_0 = s, t) = 1$$
$$F(v_i, t) = \sum_{(\{v_{i-1}, v_i\}, t') \in \mathcal{E} \text{ with } t' \leq t} F(v_{i-1}, t') \text{ for } i \geq 1.$$

It is straightforward to check that $F(z, T)$ can be computed in the claimed running time. We now formally prove correctness by induction on $i$. That is, we prove that $F(v_i, t)$ equals the number of temporal $(s, v_i)$-paths that start at $s$ and arrive at $v_i$ at time $t$ or earlier; it will follow immediately that $F(z, T)$ is the number of $(s, z)$-paths, so it suffices to compute $F(v_i, t)$ for all $0 \leq i \leq |P|$.

The base case $i = 0$ is trivial. Assume that $i > 0$. We sum over the last time-edge of the temporal $(s, v_i)$-paths starting at $s$ and arriving at $v_i$ at time $t$ or earlier. Let $\mathcal{P}$ be the set of all temporal $(s, v_i)$-paths starting at $s$ and arriving at $v_i$ at time $t$ or earlier that use $(\{v_{i-1}, v_i\}, t') \in \mathcal{E}$ as the last time-edge. All these temporal paths need to arrive at $v_{i-1}$ at time $t'$ or earlier, otherwise they cannot use time-edge $(\{v_{i-1}, v_i\}, t')$. Since all temporal paths in $\mathcal{P}$ do not differ in the last time-edge, the cardinality of $\mathcal{P}$ equals the number of

temporal $(s, v_{i-1})$-paths starting at $s$ and arriving at $v_{i-1}$ at time $t'$ or earlier. By the induction hypothesis this number equals $F(v_{i-1}, t')$. Clearly, if the last time-edge of two temporal $(s, v_i)$-paths starting at $s$ and arriving at $v_i$ at time $t$ or earlier is different, then the two temporal paths are different, so we do not double count. Hence, we have shown that $F(v_i, t)$ equals the number of temporal $(s, v_i)$-paths that start at $s$ and arriving at $v_i$ at time $t$ or earlier. ◀

## 4.2 Generalisations of the Forest Algorithm

In this subsection, we present two generalisations of Theorem 8. The first one results in an FPT-algorithm for the timed-feedback vertex number as a parameter and the second one in an FPT-algorithm for the feedback edge number of the underlying graph as a parameter. We remark that both parameters are larger than the feedback vertex number of the underlying graph, for which Theorem 5 refutes the existence of FPT-algorithms. Both algorithms are inspired by algorithms presented by Casteigts et al. [13] for the so-called Restless Temporal Path problem.

The timed feedback vertex number was introduced by Casteigts et al. [13] and, intuitively, counts the minimum number of *vertex appearances* that need to be removed from a temporal graph to make its underlying graph cycle-free. Formally, it is defined as follows.

▶ **Definition 9** ([13]). *Let $\mathcal{G} = (V, \mathcal{E}, T)$ be a temporal graph. A* timed feedback vertex set *of $\mathcal{G}$ is a set $X \subseteq V \times [T]$ of vertex appearances such that the underlying graph of $\mathcal{G}' = (V, \mathcal{E}', T)$ is a forest, where $\mathcal{E}' := \mathcal{E} \setminus \{(\{v, w\}, t) \in \mathcal{E} \mid (v, t) \in X \vee (w, t) \in X\}$. The* timed feedback vertex number *of a temporal graph $\mathcal{G}$ is the minimum cardinality of a timed feedback vertex set of $\mathcal{G}$.*

Our FPT-algorithm for the timed feedback vertex number as a parameter follows similar ideas as the one by Casteigts et al. [13] for the Restless Temporal Path problem. Roughly speaking, our algorithm performs the following steps.

1. Compute a minimum cardinality timed feedback vertex set of the input temporal graph.
2. Iterate over all possibilities for how a temporal path can traverse the vertex appearances in the timed feedback vertex set.
3. For each possibility, create an instance of the so-called #Weighted Multicoloured Independent Set on Chordal Graphs problem to compute the number of possibilities for connecting the vertex appearances of the timed feedback vertex set that are supposed to be traversed.
4. Use this to compute the total number of temporal $(s, z)$-paths in the temporal input graph.

The intuition here is that the possibilities for connecting the vertex appearances of the timed feedback vertex set that are supposed to be traversed correspond to path segments in the underlying graph of the temporal graph without the timed feedback vertex set, which is a forest. It is well-known that chordal graphs are intersection graphs of subtrees in forest [25]. This means that an independent set in a chordal graph corresponds to a selection of non-intersecting subtrees (which here will all be paths). The colours can be used to make sure that, for each pair of vertex appearances of the timed feedback vertex set that are supposed to be traversed directly after one another, exactly one path segment connecting them can be in the independent set. The weights can be used to model how many temporal paths follow the corresponding path segment of the underlying graph.

As mentioned above, we have to solve #Weighted Multicoloured Independent Set on Chordal Graphs as a subroutine instead of the unweighted decision version of the problem. This is the main difference between our algorithm and the one by Casteigts et al. [13]. In the following we give a formal definition.

| #Weighted Multicoloured Independent Set on Chordal Graphs | |
|---|---|
| Input: | A chordal graph $G = (V, E)$, a colouring function $c : V \to [k]$, and a weight function $w : V \to \mathbb{N}$. |
| Task: | Compute $\sum_{X \subseteq V \mid X \text{ is a multicoloured independent set in } G} \prod_{v \in X} w(v)$. |

We can observe that #Weighted Multicoloured Independent Set on Chordal Graphs presumably cannot be solved in polynomial time. This follows directly from the NP-hardness of Multicoloured Independent Set on Chordal Graphs [7, Lemma 2]. Hence, we have the following.

▶ **Observation 10.** *#Weighted Multicoloured Independent Set on Chordal Graphs cannot be solved in polynomial time unless P=NP.*

However, we can obtain an FPT-algorithm for #Weighted Multicoloured Independent Set on Chordal Graphs parameterised by the number of colours. This will be sufficient for our purposes. To show this result, we adapt an algorithm by Bentert et al. [6, Proposition 5.6] to solve Maximum Weight Multicoloured Independent Set on Chordal Graphs, where given a chordal graph $G = (V, E)$, a colouring function $c : V \to [k]$, and a weight function $w : V \to \mathbb{N}$, one is asked to compute a multicoloured independent set of maximum weight in $G$. Here, the weight of an independent set is the *sum* of the weights of its vertices. Note that in #Weighted Multicoloured Independent Set on Chordal Graphs the weight of an independent set is the *product* of the weights of its vertices.

▶ **Proposition 11** (⋆). *#Weighted Multicoloured Independent Set on Chordal Graphs is fixed-parameter tractable when parameterised by the number $k$ of colours.*

Using Proposition 11, we are ready to give our FPT-algorithm for #Temporal Path parameterised by the timed feedback vertex number.

▶ **Theorem 12** (⋆). *#Temporal Path is fixed-parameter tractable when parameterised by the timed feedback vertex number of the input temporal graph.*

Now we consider the feedback edge number of the input temporal graph as our parameter, and show the following fixed-parameter tractability result. It is very similar to an algorithm by Casteigts et al. [13] for the so-called Restless Temporal Path problem parameterised by the feedback edge number, we only sketch the proof.

▶ **Theorem 13.** *#Temporal Path is fixed-parameter tractable when parameterised by the feedback edge number of the underlying graph of the input temporal graph.*

**Proof Sketch.** Let $(\mathcal{G}, s, z)$ be an instance of #Temporal Path. We adapt an algorithm by Casteigts et al. [13, Theorem 7] for the so-called Restless Temporal Path problem. The algorithm consist of four steps (only the last step needs adaptation to our problem):

1. Exhaustively remove vertices with degree $\leq 1$ from the underlying graph of $\mathcal{G}$ (except $s$ and $z$). Let $G'$ be the resulting (static) graph.
2. Compute a minimum feedback edge set $F$ of $G'$. Let $f := |F|$.
3. Let $V^{\geq 3}$ denote all vertices of $G'$ with degree at least three. Partition the forest $G' - F$ into a set of maximal paths $\mathcal{P}$ with endpoints in $\bigcup_{e \in F} e \cup V^{\geq 3} \cup \{s, z\}$, and intermediate vertices all of degree 2. It holds that $|\mathcal{P}| \in \mathcal{O}(f)$ [4, Lemma 2].
4. Any temporal $(s, z)$-path in $\mathcal{G}$ can be formed with time-edges whose underlying edges are feedback edges from $F$ or form paths in $\mathcal{P}$. Enumerate all $2^{\mathcal{O}(f)}$ sequences of underlying edges that a temporal $(s, z)$-path in $\mathcal{G}$ can follow and for each one count the temporal $(s, z)$-paths following these underlying edges using Theorem 8. Add up all path counts.

The correctness follows from the correctness of [13, Theorem 7] and that due to the exhaustive search, all temporal $(s, z)$-paths in $\mathcal{G}$ are considered and correctly counted. ◀

## 4.3 Parameterisation by Treewidth and Lifetime

Our goal in this subsection is to demonstrate that #Temporal Path is in FPT when parameterised simultaneously by the treewidth of the underlying graph and the lifetime; to do this we give an MSO-encoding of the problem and make use of the counting version of Courcelle's theorem for model-checking on relational structures [15].

▶ **Theorem 14** (⋆). *#Temporal Path is in FPT when parameterised by the combination of the treewidth of the underlying graph and the lifetime.*

## 4.4 Parameterisation by Vertex-Interval-Membership-Width

In this subsection, we present an FPT algorithm for #Temporal Path parameterised by the so-called vertex-interval-membership-width of the input temporal graph. The vertex-interval-membership-width is a temporal graph parameter recently introduced by Bumpus and Meeks [11] which, like the timed feedback vertex number, depends not only on the structure of the underlying graph but also on the assignment of times to edges. Intuitively, the vertex-interval-membership-width counts the maximum number of vertices that are "relevant" at any timestep, where a vertex is considered relevant if it has an incident edge both (weakly) before and after the current timestep (so, for example, a vertex $v$ is relevant only at times when a temporal path could have entered but not yet left $v$).

▶ **Definition 15** ([11]). *The* vertex interval membership sequence *of a temporal graph $(G, \mathcal{E}, T)$ is the sequence $(F_t)_{t\in[T]}$ of vertex-subsets of $G$ where*

$$F_t := \{v \in V(G) \mid \exists i \leq t \leq j \text{ and } u, w \in V(G) \text{ such that } \{u, v\} \in E_i \text{ and } \{w, v\} \in E_j\}.$$

*Note that we allow $u = w$. The* vertex-interval-membership-width *of $(G, \mathcal{E}, T)$ is the integer* $\mathrm{vimw}(G, \mathcal{E}, T) := \max_{t\in[T]} |F_t|$.

Note that every vertex incident with an edge in $E_i$ must belong to $F_i$, and so $|E_i| \leq \binom{|F_i|}{2} \leq |F_i|^2$. The vertex interval membership sequence gives us a structure we can use for dynamic programming, which we exploit to obtain the following result.

▶ **Theorem 16** (⋆). *#Temporal Path can be solved in time $\mathcal{O}(w^{2w^2+w} \cdot T)$ where $T$ and $w$ are the lifetime and vertex-interval-membership-width respectively of the input graph.*

In our dynamic programming algorithm, a state of the bag $F_t$ is a pair $(v, X)$, where $v \in F_t$ and $X \subseteq F_t \setminus \{v\}$. For any state $(v, X)$ of $F_t$, we compute the number $P_t(v, X)$ of temporal paths $Q$ from $s$ to $v$, arriving by time $t$, such that $V(Q) \cap (F_t \setminus \{v\}) = X$. Computing all such values $P_t(v, X)$ is clearly sufficient, since the total number of temporal $(s, z)$-paths is $\sum_{Y\subseteq F_T\setminus\{z\}} P_T(z, Y)$. We compute the values for each bag $F_t$ in turn, assuming for $t \geq 1$ that we have already computed all counts corresponding to $F_{t-1}$.

# 5 Approximation Algorithms for Temporal Path Counting

In this section we consider the problems of approximating #Temporal Path and approximating the temporal betweenness centrality. For #Temporal Path, recall from Section 3.2 that there is unlikely to be an FPRAS for #Temporal Path in general; in Section 5.1, we show that there is however an FPTRAS for #Temporal Path when the maximum permitted path length is taken as the parameter. This in turn implies the existence of an FPTRAS for #Temporal Path when restrictions are placed on the structure of the

underlying graph that limit the length of the longest path. We remark that Theorem 5 and Theorem 6 do not rule out exact FPT-algorithms for these parameterisations. We leave open whether stronger hardness results or exact algorithms for this case can be obtained.

In Section 5.2 we apply this approximation result to the problem of approximating temporal betweenness: we demonstrate that, whenever we can efficiently approximate #Temporal Path, we can efficiently estimate the maximum temporal betweenness centrality over all vertices of the input graph.

## 5.1 Approximately Counting Short Temporal Paths

In this subsection we consider the complexity of approximately counting $(s, z)$-paths parameterised by the length of the path.

| #Short Temporal Path | |
|---|---|
| Input: | A temporal graph $\mathcal{G} = (V, \mathcal{E}, T)$, two vertices $s, z \in V$, and an integer $k$. |
| Task: | Count the temporal $(s, z)$-paths in $\mathcal{G}$ that contain exactly $k$ edges. |

We prove the following result.

▶ **Theorem 17.** *There is a randomised algorithm which, given as input an instance $(\mathcal{G}, s, z)$ of #Short Temporal Path together with error parameters $\varepsilon > 0$ and $0 < \delta < 1$, outputs an estimate $\hat{N}$ of the number of temporal $(s, z)$-paths in $\mathcal{G}$ containing exactly $k$ edges; with probability at least $1-\delta$, $\hat{N}$ is an $\varepsilon$-approximation to the number of $(s, z)$-paths in $\mathcal{G}$ containing exactly $k$ edges. The running time of the algorithm is $\mathcal{O}(k!e^k \log(1/\delta)\varepsilon^{-2}n^2T^2)$.*

The key ingredient in the proof is an efficient algorithm for the *multicoloured* version of this problem, in which the input graph is equipped with a vertex-colouring (not necessarily proper) and we wish to count paths containing exactly one vertex of each colour. We note that #Short Temporal Path meets the conditions for a *uniform witness problem* given by Dell et al. [17], and therefore in order to demonstrate the existence of an FPTRAS it would suffice to demonstrate that the multicoloured version of the problem admits an FPT decision algorithm. However, in this case it is easy to show that in fact *exact* counting is tractable in the multicoloured setting, so we can infer the existence of an FPTRAS immediately by applying a standard colour-coding technique, without invoking the power of the metatheorem by Dell et al. [17].

| #Multicoloured Temporal Path | |
|---|---|
| Input: | A temporal graph $\mathcal{G} = (V, \mathcal{E}, T)$, two vertices $s, z \in V$, and a partition of $V \setminus \{s, z\}$ into colour sets $V_1 \uplus \cdots \uplus V_\ell$. |
| Task: | Count the number of temporal $(s, z)$-paths that contain exactly one vertex from each colour-set $V_1, \ldots, V_\ell$. |

▶ **Proposition 18** ($\star$). *#Multicoloured Temporal Path is solvable in $\mathcal{O}((\ell+1)!n^2T^2)$ time.*

Equipped with this algorithm for #Multicoloured Temporal Path, we use a standard colour-coding technique to obtain an FPTRAS for #Short Temporal Path. This involves repeatedly generating random colourings (not necessarily proper) of the vertices of $V \setminus \{s, z\}$ using $k-1$ colours; note that a single colouring can clearly be generated in time $\mathcal{O}(nk)$. For each colouring, we solve the corresponding instance of #Multicoloured Temporal Path using the algorithm of Proposition 18. Setting $N$ to be the sum of counts over all

colourings, we return $Nk^k/k!$. Following the argument by Alon et al. [1, Section 2.1], we see that the number of colourings we must generate to obtain an $\varepsilon$-approximation to #Short Temporal Path with probability at least $1-\delta$ is $\mathcal{O}(e^k \log(1/\delta)\varepsilon^{-2})$, giving the result.

Since the maximum possible path length is bounded by a function of either the vertex cover number or the treedepth[4] of the underlying input graph, we immediately obtain the following corollary to Theorem 17.

▶ **Corollary 19.** *#Temporal Path admits an FPTRAS parameterised by either vertex cover number or treedepth of the underlying input graph.*

## 5.2 Approximating Temporal Betweenness

Observe that it is not clear how to use an approximation algorithm for #Temporal Path to approximate the temporal betweenness centrality for every vertex in the input graph (we give a detailed discussion in Section 2.3). In this section, we address the simpler problem of determining the maximal temporal betweenness centrality of any vertex in the graph: we show that we can efficiently approximate this quantity whenever there is an FPRAS (or FPTRAS) for #Temporal Path.

▶ **Theorem 20** ($\star$)**.** *Let $\mathcal{C}$ be a class of temporal graphs on which #Temporal Path admits an FPRAS. Then $\mathcal{C}$ admits an FPRAS to estimate, given an input temporal graph $\mathcal{G} = (V, \mathcal{E}, T) \in \mathcal{C}$, $\max_{v \in V} C_B^{(\star)}(v)$, for $\star \in \{$fastest, foremost$\}$. Similarly, if $\mathcal{C}$ is a class of graphs on which there exists an FPTRAS for #Temporal Path with respect to some parameterisation $\kappa$ then, with respect to the same parameterisation, $\mathcal{C}$ admits an FPTRAS to estimate, given an input temporal graph $\mathcal{G} = (V, \mathcal{E}, T) \in \mathcal{C}$, $\max_{v \in V} C_B^{(\star)}(v)$, for $\star \in \{$fastest, foremost$\}$.*

The proof relies on the fact that we may assume that at least one vertex has temporal betweenness centrality at least $\frac{1}{n(T+1)}$, where $n$ is the number of vertices; we begin by arguing that we can efficiently identify the inputs for which this lower bound does not hold, and that in these cases the correct answer is in fact 0. Using this assumption, we show that the following procedure is likely to produce a good approximation to $\max_{v \in V} C_B^{(\star)}(v)$: for each vertex pair $(s, z)$, sample a large (polynomial) number of $\star$-optimal temporal $(s, z)$-paths, and record the number that contain each vertex $v$ as an internal vertex; after considering all pairs $(s, z)$, we assume that the vertex $v_{\max}$ we have seen most frequently has the maximum betweenness centrality, and return as our estimate the proportion of sampled paths that contain $v_{\max}$. We note that, applying a general result of Jerrum et al. [30], we can assume the existence of an efficient algorithm to sample $\star$-optimal temporal $(s, z)$-paths almost uniformly whenever there is an FPRAS (or FPTRAS).

Combining Theorem 20 with Corollary 19 gives the following immediate corollary.

▶ **Corollary 21.** *There is an FPTRAS which, given as input a temporal graph $\mathcal{G} = (V, \mathcal{E}, T)$, computes an approximation to $\max_{v \in V} C_B^{(\star)}(v)$ (for $\star \in \{$fastest, foremost$\}$), parameterised by either the vertex cover number or treedepth of the underlying input graph.*

[4] We refer to the book of Nesetril and de Mendez [37] for the definition of treedepth, and a proof that the maximum length of a path in a graph is bounded by a function of its treedepth.

## 6 Conclusion

In this work, we initiate the systematic study of the parameterised and approximation complexity of #Temporal Path. We present parameterised and approximation hardness results and complement them with several parameterised exact and approximation algorithms.

In terms of improving our results, we conjecture that it is possible to prove #W[1]-hardness instead of $\oplus$W[1]-hardness for #Temporal Path parameterised by the feedback vertex number of the underlying graph. Furthermore, we leave open whether our parameterised approximation results for vertex cover number or treedepth of the underlying graph can be improved from a classification standpoint by obtaining exact algorithms, or whether we can also show parameterised hardness for those cases.

We leave open to what extent our results transfer to the problem of counting *strict* temporal $(s, z)$-paths, where the labels on the time-edges have to be strictly increasing. We conjecture that most of our results hold for the strict case. In fact, we believe that the MSO formulation used to obtain fixed-parameter tractability for the treewidth of the underlying graph combined with the lifetime can be simplified: in the strict case, a first-order formula should suffice, which would lead to fixed-parameter tractability in terms of the lifetime on any class of nowhere-dense graphs [26], or for the combined parameter of cliquewidth and lifetime [14].

We conjecture that our polynomial-time algorithm for the case where the underlying graph is a forest (Section 4.1) can be extended to the case where the underlying graph is series-parallel [20]. Recall that a forest can be seen as a series-parallel graph where only series compositions are used. The idea would be to extend the dynamic program to parallel compositions, exploiting the observation that any temporal path can visit the terminal vertices of a series-parallel graph at most once.

Finally, we believe that our FPT-algorithms presented in Section 4.2 for the timed feedback vertex number and feedback edge number of the underlying graph, respectively, can be adapted to count other types of temporal $(s, z)$-paths for which counting is in general #P-hard. A key ingredient for both algorithms is a polynomial-time algorithm for instances that have a forest as underlying graph. This leads us to believe that the algorithms can be modified to count restless temporal $(s, z)$-paths [13] and possibly also to count delay-robust $(s, z)$-routes [24], since both of these path types can be found in polynomial time when the underlying graph of the input temporal graph is a forest.

## References


1 Noga Alon, Phuong Dao, Iman Hajirasouliha, Fereydoun Hormozdiari, and Süleyman Cenk Sahinalp. Biomolecular network motif counting and discovery by color coding. In *Proceedings 16th International Conference on Intelligent Systems for Molecular Biology (ISMB '08)*, pages 241–249, 2008.

2 Ahmad Alsayed and Desmond J Higham. Betweenness in time dependent networks. *Chaos, Solitons & Fractals*, 72:35–48, 2015.

3 Vikraman Arvind and Venkatesh Raman. Approximation algorithms for some parameterized counting problems. In Prosenjit Bose and Pat Morin, editors, *Proceedings of the 13th International Symposium on Algorithms and Computation (ISAAC '02)*, volume 2518 of *Lecture Notes in Computer Science*, pages 453–464. Springer, 2002.

4 Matthias Bentert, Alexander Dittmann, Leon Kellerhals, André Nichterlein, and Rolf Niedermeier. An adaptive version of Brandes' algorithm for betweenness centrality. *Journal of Graph Algorithms and Applications*, 24(3):483–522, 2020.

5 Matthias Bentert, Anne-Sophie Himmel, André Nichterlein, and Rolf Niedermeier. Efficient computation of optimal temporal walks under waiting-time constraints. *Applied Network Science*, 5(1):73, 2020.

6 Matthias Bentert, René van Bevern, and Rolf Niedermeier. Inductive $k$-independent graphs and $c$-colorable subgraphs in scheduling: a review. *Journal of Scheduling*, 22(1):3–20, 2019.

7 René Bevernvan Bevern, Matthias Mnich, Rolf Niedermeier, and Mathias Weller. Interval scheduling and colorful independent sets. *Journal of Scheduling*, 18(5):449–469, 2015.

8 Andreas Björklund, Holger Dell, and Thore Husfeldt. The parity of set systems under random restrictions with applications to exponential time problems. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP '15)*, pages 231–242. Springer, 2015.

9 Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

10 Binh-Minh Bui-Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285, 2003.

11 Benjamin Merlin Bumpus and Kitty Meeks. Edge exploration of temporal graphs. In *Proceedings of the 32nd International Workshop on Combinatorial Algorithms (IWOCA '21)*, volume 12757 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2021.

12 S. Buß, H. Molter, R. Niedermeier, and M. Rymar. Algorithmic aspects of temporal betweenness. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20)*, pages 2084–2092, 2020. `arXiv:2006.08668`.

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

14 B. Courcelle, J.A. Makowsky, and U. Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics*, 108(1):23–52, 2001.

15 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2012.

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

17 Holger Dell, John Lapinskas, and Kitty Meeks. Approximately counting and sampling small witnesses using a colourful decision oracle. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '20)*, pages 2201–2180. Society for Industrial and Applied Mathematics, 2020.

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

19 Jessica A. Enright, Kitty Meeks, and Hendrik Molter. Counting temporal paths. *CoRR*, abs/2202.12055, 2022. `arXiv:2202.12055`.

20 David Eppstein. Parallel recognition of series-parallel graphs. *Information and Computation*, 98(1):41–55, 1992.

21 Jörg Flum and Martin Grohe. The parameterized complexity of counting problems. *SIAM Journal on Computing*, 33(4):892–922, 2004.

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

23 Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

24 Eugen Füchsle, Hendrik Molter, Rolf Niedermeier, and Malte Renken. Delay-robust routes in temporal graphs. In *Proceedings of the 39th International Symposium on Theoretical Aspects of Computer Science (STACS '22)*, volume 219 of *LIPIcs*, pages 30:1–30:15, 2022.

25 Fănică Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.

26 Martin Grohe and Nicole Schweikardt. First-order query evaluation with cardinality conditions. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (SIGMOD/PODS '18)*, pages 253–266. ACM, 2018.

27 Habiba, Chayant Tantipathananandh, and Tanya Y Berger-Wolf. Betweenness centrality measure in dynamic networks. Technical Report 19, Department of Computer Science, University of Illinois at Chicago, Chicago, 2007. DIMACS Technical Report.

28 Petter Holme. Modern temporal network theory: a colloquium. *The European Physical Journal B*, 88(9):234:1–234:30, 2015.

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

30 Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

31 Minghui Jiang. On the parameterized complexity of some optimization problems related to multiple-interval graphs. *Theoretical Computer Science*, 411(49):4253–4262, 2010.

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

33 Hyoungshick Kim and Ross Anderson. Temporal node centrality in complex networks. *Physical Review E*, 85(2):026107, 2012.

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

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

36 Petra Mutzel and Lutz Oettershagen. On the enumeration of bicriteria temporal paths. In *Proceedings of the 15th Annual Conference on Theory and Applications of Models of Computation (TAMC '19)*, volume 11436 of *Lecture Notes in Computer Science*, pages 518–535. Springer, 2019.

37 Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and Combinatorics*. Springer, 2012.

38 Vincenzo Nicosia, John Tang, Cecilia Mascolo, Mirco Musolesi, Giovanni Russo, and Vito Latora. Graph metrics for temporal networks. In *Temporal Networks*, pages 15–40. Springer, 2013.

39 Amir Afrasiabi Rad, Paola Flocchini, and Joanne Gaudet. Computation and analysis of temporal betweenness in a knowledge mobilization network. *Computational Social Networks*, 4(1):5, 2017.

40 Maciej Rymar, Hendrik Molter, André Nichterlein, and Rolf Niedermeier. Towards classifying the polynomial-time solvability of temporal betweenness centrality. In *Proceedings of the 47th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '21)*, volume 12911 of *Lecture Notes in Computer Science*, pages 219–231. Springer, 2021.

41 Frédéric Simard, Clémence Magnien, and Matthieu Latapy. Computing betweenness centrality in link streams. *CoRR*, abs/2102.06543, 2021. `arXiv:2102.06543`.

42 Alistair Sinclair. *Randomised algorithms for counting and generating combinatorial structures*. PhD thesis, University of Edinburgh, 1988.

43 John Tang, Ilias Leontiadis, Salvatore Scellato, Vincenzo Nicosia, Cecilia Mascolo, Mirco Musolesi, and Vito Latora. Applications of temporal graph metrics to real-world networks. In *Temporal Networks*, pages 135–159. Springer, 2013.

44 John Tang, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Vincenzo Nicosia. Analysing information flows and key mediators through temporal centrality metrics. In *Proceedings of the 3rd ACM Workshop on Social Network Systems*, pages 3:1–3:6. ACM, 2010.

45 Ioanna Tsalouchidou, Ricardo Baeza-Yates, Francesco Bonchi, Kewen Liao, and Timos Sellis. Temporal betweenness centrality in dynamic graphs. *International Journal of Data Science and Analytics*, 9(3):257–272, 2020.

46 Leslie G Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

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