How fast can we reach a target vertex in stochastic temporal graphs?

Eleni C. Akrida
Department of Computer Science, University of Liverpool, UK
eleni.akrida@liverpool.ac.uk

George B. Mertzios
Department of Computer Science, Durham University, UK
george.mertzios@durham.ac.uk

Sotiris Nikoletseas
Computer Engineering & Informatics Department, University of Patras, and CTI, Greece
nikole@cti.gr

Christoforos Raptopoulos
Computer Engineering & Informatics Department, University of Patras, and CTI, Greece
raptopox@ceid.upatras.gr

Paul G. Spirakis
Department of Computer Science, University of Liverpool, UK
Computer Engineering & Informatics Department, University of Patras, Greece
p.spirakis@liverpool.ac.uk

Viktor Zamaraev
Department of Computer Science, Durham University, UK
viktor.zamaraev@durham.ac.uk

Abstract

Temporal graphs are used to abstractly model real-life networks that are inherently dynamic in nature, in the sense that the network structure undergoes discrete changes over time. Given a static underlying graph $G = (V, E)$, a temporal graph on G is a sequence of *snapshots* $\{G_t = (V, E_t) \subseteq G : t \in \mathbb{N}\}$, one for each time step $t \geq 1$. In this paper we study *stochastic temporal graphs*, i.e. stochastic processes $\mathcal{G} = \{G_t \subseteq G : t \in \mathbb{N}\}$ whose random variables are the snapshots of a temporal graph on G. A natural feature of stochastic temporal graphs which can be observed in various real-life scenarios is a *memory effect* in the appearance probabilities of particular edges; that is, the probability an edge $e \in E$ appears at time step t depends on its appearance (or absence) at the previous k steps. In this paper we study the hierarchy of models *memory-k*, $k \geq 0$, which address this memory effect in an *edge-centric* network evolution: every edge of G has its own probability distribution for its appearance over time, *independently* of all other edges. Clearly, for every $k \geq 1$, memory-$(k-1)$ is a special case of memory-k. However, in this paper we make a clear distinction between the values $k = 0$ (*"no memory"*) and $k \geq 1$ (*"some memory"*), as in some cases these models exhibit a fundamentally different computational behavior for these values of k, as our results indicate. For every $k \geq 0$ we investigate the computational complexity of two naturally related, but fundamentally different, *temporal path* (or *journey*) problems: Minimum Arrival and Best Policy. In the first problem we are looking for the *expected arrival time* of a foremost journey between two designated vertices s, y. In the second one we are looking for the expected arrival time of the *best policy* for actually choosing a *particular* s-y journey. We present a detailed investigation of the computational landscape of both problems for the different values of memory k. Among other results we prove that, surprisingly, Minimum Arrival is *strictly harder* than Best Policy; in fact, for $k = 0$, Minimum Arrival is #P-hard while Best Policy is solvable in $O(n^2)$ time.

2012 ACM Subject Classification Mathematics of computing → Graph theory; Mathematics of computing → Graph algorithms; Mathematics of computing → Paths and connectivity problems

Keywords and phrases Temporal network, stochastic temporal graph, temporal path, #P-hard problem, polynomial-time approximation scheme.

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 126; pp. 126:1–126:15
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.126

Related Version A full version of the paper is available at `https://arxiv.org/abs/1903.03636` [2].

Funding This work was supported by the NeST initiative of the EEE/CS School of the University of Liverpool and by the EPSRC grants EP/P020372/1 and EP/P02002X/1.

1 Introduction

Dynamic network analysis, i.e. analysis of networks that change over time, is currently one of the most active topics of research in network science and theory. A common task in this field is to use our prior knowledge of the network link dynamics to answer questions about the behavior of the network over time, e.g. how quickly information can flow through it. Many modern real-life networks are dynamic in nature, in the sense that the network structure undergoes discrete changes over time [31, 36]. Here we deal with the discrete-time dynamicity of the network links (edges) over a fixed set of nodes (vertices). That is, given an underlying static graph G, the network evolution over G is given by the successive appearance or absence of each edge of G at every time step $t = 1, 2, \ldots$. This concept of dynamic network evolution is given by *temporal graphs* [27, 29], which are also known by other names such as *evolving graphs* [6, 20], or *time-varying graphs* [1]. For a recent attempt to integrate existing models, concepts, and results from the distributed computing perspective, see the survey papers [12, 13] and the references therein.

▶ **Definition 1** (Temporal graph). *Given an underlying static graph $G = (V, E)$ on n vertices and m edges, a* temporal graph *on G is a sequence $\mathcal{G} = \{G_t = (V, E_t) : t \in \mathbb{N}\}$ of graphs such that $E_t \subseteq E$ for all $t \in \mathbb{N}$. Every G_t is the* snapshot *of $\mathcal{G}$ at time step t.*

Another way to think about temporal graphs is by assigning *time-labels* on the edges; for example, if an edge e appears in the snapshots G_3, G_5, and G_8, then we equivalently assign to e the set of labels $\lambda(e) = \{3, 5, 8\}$. Due to the vast applicability of temporal graphs, various structural and algorithmic properties of them have been studied extensively, both via theoretical/algorithmic analysis and via empirical simulation-based analysis. In many of these works, one of the central temporal notions is that of a temporal path. A path in the underlying (static) graph G is a *temporal path* (or *journey*) if there exists an increasing sequence of time-labels as one walks along the edges of the path [27, 29]. Motivated by the fact that, due to causality, information in temporal graphs can only flow along sequences of edges that appear in an increasing time order, many temporal graph parameters and optimization problems that have been studied so far are based on the notion of a temporal path and other related notions, e.g. temporal analogs of distance, diameter, connectivity, reachability, and exploration [3, 4, 7, 8, 10, 14, 18, 19, 21, 23, 28, 33]. In addition to temporal paths, recently also various temporal non-path problems have been introduced and algorithmically studied, such as temporal vertex cover [5], temporal coloring [30], and temporal Δ-cliques [24, 38].

Apart from the focus on the various algorithmic problems that one can study on temporal graphs, one can also view temporal graphs through several different levels of knowledge about the actual network evolution. On the one extreme, we may be given the whole temporal graph instance in advance, i.e. the times of appearance and absence of every edge at all times, as it typically happens e.g. when modeling transportation networks. On the other extreme, the temporal graph may be created by an adversary who reveals it to us snapshot-by-snapshot at every time step. Here we focus on the intermediate knowledge settings, captured by *stochastic temporal graphs*, where the network evolution is given by a probability distribution that governs the appearance of each edge over time.

▶ **Definition 2** (Stochastic temporal graph)**.** *A* stochastic temporal graph *is a stochastic process $\mathcal{G} = \{G_t : t \in \mathbb{N}\}$ whose random variables are snapshots $G_t \subseteq G$ of an underlying graph G. Every instantiation of $\mathcal{G}$ is a temporal graph.*

A natural feature of stochastic temporal graphs which can be observed in various real-life scenarios (and which we address in this paper) is that the appearance probability of a particular edge at a given time step t depends on the appearance (or absence) of the same edge at the previous $k \geq 1$ time steps. This "memory effect" can often be observed, among others, in faulty network communication and in mobile, social, and peer-to-peer networks [15, 34, 37]. Several other models of temporal networks which exhibit some sort of probabilistic behavior have been considered in the past, see e.g. [25].

In this paper, we study a hierarchy of models for stochastic temporal graphs which address an *edge-centric* network evolution, i.e. they assign to every edge of the underlying graph G a probability distribution for its appearance over time, independently of all the other edges. The first and most basic model (*memoryless* or *memory*-0) assigns independently to every edge e a probability p_e such that, at every time step, e appears with probability p_e. In the general model (*memory-k*), at every time step the appearance probability of every edge is a function of the history of its appearances/absences in the last $k \geq 1$ time steps. Clearly, for every $k \geq 1$, the memory-$(k-1)$ model is a special case of the memory-k model. However, in this paper we make a clear distinction between the values $k = 0$ (*"no memory"*) and $k \geq 1$ (*"some memory"*), as in some cases these models exhibit a fundamentally different computational behavior for these values of k, as our results indicate (see Section 4).

Our memory-k model, $k \geq 1$, is a direct generalization of the homogeneous version of the memory-1 model that was introduced in a seminal paper by Clementi et al. [16], in which all edges have the same probability distribution for their appearance, based on their own appearance/absence at the previous step. In this homogeneous memory-1 model, Clementi et al. gave upper bounds for the flooding time and they provided tight characterizations of the graphs on which the flooding time is constant [16]. It is worth noting here that Avin et al. [7] studied the completely opposite extreme of our edge-centric evolution; namely they considered a *graph-centric* evolution model where a global probability distribution assigns specific transition probabilities among different snapshots [7]. Between the two extremes of the edge-centric and the graph-centric network evolution models, there exists a whole hierarchy of locally interdependent probabilistic patterns, i.e. probability distributions where the appearance probability of one edge also depends on the appearance of *other edges* over time; such models remain mostly unexplored.

In both our memoryless and memory-k variations of stochastic temporal graphs, we study two fundamental temporal path (i.e. journey) problems that are defined on two designated vertices s and y. Consider a piece of information that is generated at s at time 1, which we would like to send to y via an s-y journey. The *arrival time* of an s-y journey in a realization of a stochastic temporal graph is the time the information reaches y using this journey. A *foremost s-y* journey is one with the smallest arrival time. In the first part of the paper we investigate the complexity of computing the *expected arrival time* of a *foremost s-y* journey. Basu et al. [9] and Nain et al. [32] studied a similar problem but their work is restricted to the simpler cases where the underlying graph is either a path or a grid.

In the second part of the paper we investigate the complexity of computing the arrival time of a *best policy* for actually choosing a particular s-y journey in the stochastic temporal graph. To illustrate this notion of a best policy, assume that some piece of information is carried by an entity, say Alice. Alice is given as input the parameters of the stochastic temporal graph (i.e. the probabilistic rules on the edges) and, at every time step, she knows the current snapshot and her current location. Based on this information, Alice has to decide at every step for her next action, while her goal is to reach y as quickly as possible on expectation, starting at time 1. In a very inspiring paper, Basu et al. [8] consider this problem in the special case of the memoryless model where all edges have the same probability of appearance at every time, and give a Dijkstra-like polynomial-time algorithm. Special cases of the memory-1 model were considered in [11].

To illustrate the difference between the two problems we study, we make the following analogy. In the first problem (Minimum Arrival) we try to transfer information from s to y using an unbounded number of messages, i.e. we "flood" the stochastic temporal graph with information. Initially the information is stored at s at time 1 and then, at every step, every informed vertex informs all its neighbors as soon as the edge between them becomes available. In the second problem (Best Policy) we try to transfer a package with a tangible good from s to y. Now, at every step we need to decide for the actual route of the package through the network: when an edge appears, should we ship the package along it or rather wait where we currently are? Best Policy is more relevant to real-life applications than Minimum Arrival, where an actual *good* journey needs to be found in real time.

Our contribution. In the first part of the paper, in Section 3, we provide our results for the problem Minimum Arrival, i.e. for computing the expected arrival time of a foremost s-y journey in a stochastic temporal graph. First we prove in Section 3.1 that Minimum Arrival is #P-hard even for the memoryless model (and thus also for the memory-k model, for every $k \geq 1$). The reduction is done from the problem #PP2DNF which counts the number of satisfying assignments in a positive partitioned 2-DNF Boolean formula [35].

Second, we provide in Section 3.2 a non-trivial approximation scheme for Minimum Arrival, based on dynamic programming, for the memoryless model in the case where the underlying graph G is a series-parallel graph with s and y being its terminals. More specifically, it turns out that this is a *Fully Polynomial-Time Approximation Scheme (FPTAS)* whenever the probabilities p_e are lower bounded by $\frac{1}{n^c}$ for some $c \geq 1$. Let X be the random variable that expresses the arrival time of a foremost s-y journey. For every $\varepsilon \in (0, 1]$, our FPTAS gives an algorithm that produces a value μ where $\mathbb{E}(X) - \varepsilon \leq \mu \leq \mathbb{E}(X)$, and runs in polynomial time in both n and $\frac{1}{\varepsilon}$. Although our main result of Section 3.2 concerns series-parallel graphs, we actually present a more general FPTAS approach (see Theorem 11) which is of independent interest and could lead to FPTASs also for more general classes of underlying graphs G.

Third, we present in Section 3.3 a *Fully Polynomial Randomized Approximation Scheme (FPRAS)* for Minimum Arrival in the memory-k model, for every $k \geq 0$, under the assumption that every edge appearance probability is lower bounded by $\frac{1}{n^c}$ for some $c \geq 1$. Let X be the random variable that expresses the arrival time of a foremost s-y journey. For every $\varepsilon \in (0, 1)$, our FPRAS gives a randomized algorithm that produces an estimate $\widetilde{X}$ where $(1-\varepsilon)\mathbb{E}(X) \leq \widetilde{X} \leq (1+\varepsilon)\mathbb{E}(X)$ with probability tending to 1 as $n \to \infty$, and runs in polynomial time in both n and $\frac{1}{\varepsilon}$.

In the second part of the paper, in Section 4, we provide our results for the problem Best Policy, i.e. for computing the expected arrival time of a best policy for choosing a particular s-y journey. Initially we provide in Section 4.1 a dynamic programming algorithm for the memoryless model which runs in $O(n^2)$ time and space. In wide contrast, we prove in Section 4.2 that Best Policy becomes #P-hard for the memory-k model, where $k \geq 3$, again by providing a reduction from the problem #PP2DNF. Finally, we provide in

Section 4.3 a formulation of Best Policy in the memory-k model using the general *Markov Decision Process (MDP)* framework which allows us to devise in Section 4 an exact doubly exponential-time algorithm with running time $O(2^{(kmn+n\log n)\cdot 2^{km}})$. Due to lack of space, many proofs have been omitted; the full proofs of this paper can be found in our technical report [2].

2 Preliminaries

In this paper we consider temporal graphs (see Definition 1) in which the underlying (static) graph $G = (V, E)$ has n vertices and m edges . A subgraph $H = (V, E_H)$ of G, denoted by $H \subseteq G$, is a graph where $E_H \subseteq E$. For every vertex $u \in V$, the *neighborhood* $\Gamma_G(u)$ of u in G is the set of adjacent vertices of u in G. The *closed neighborhood* $\Gamma_G[u]$ also contains vertex u itself, i.e. $\Gamma_G[u] = \Gamma_G(u) \cup \{u\}$. For simplicity of notation we denote $[n] = \{1, 2, \ldots, n\}$ for every $n \in \mathbb{N}$. Furthermore, sometimes we refer to the discrete time steps $t = 1, 2, \ldots$ as *days*. Throughout the paper we consider stochastic temporal graphs that exhibit an edge-centric evolution, i.e. every edge e of G is assigned one probability distribution for its appearance over time, independently of all other edges. We investigate the case where there is a "memory effect" that governs the probability of appearance of every edge over time. We distinguish now the cases where the the memory is zero or non-zero.

Memoryless (or memory-0) model. Every edge $e \in E$ evolves stochastically and independently of other edges as follows: at every time step $t \in \mathbb{N}$, e appears in G_t with probability p_e and is absent with probability $1 - p_e$, independently of any other time step. The numbers $\{p_e : e \in E\}$ are given parameters of the model. We denote this (memoryless) stochastic temporal graph by $\mathcal{G}^{(0)} = (G, \{p_e : e \in E\})$ or simply $\mathcal{G}^{(0)} = (G, \{p_e\})$.

Memory-k model. This model of temporal graphs exhibits stochastic time-dependency of the edges: we assume an initial (arbitrary) sequence of k snapshots, $G_{-k+1}, \ldots, G_{-1}, G_0 \subseteq G$. At every time step $t \geq 1$, every edge e appears independently of all other edges with probability that depends only on (the edge and) the history of appearance of e in the k previous snapshots. At every time step t, this history is a k-bit binary vector, where a 0-entry (resp. 1-entry) on the i-th position denotes absence (resp. appearance) of e in $E_{t-k+i-1}$, for $i = 1, \ldots, k$. Therefore the snapshot G_t is the graph that appears at time $t \geq 1$ as the result of the following experiment: given the history $H_e^{(k)}$ of the appearance of edge $e \in E$ in the last k snapshots, e belongs to E_t independently with probability $p_e(H_e^{(k)})$. We denote the memory-k stochastic temporal graph by $\mathcal{G}^{(k)}$.

In the particular case where $k = 1$, the memory-1 stochastic temporal graph $\mathcal{G}^{(1)}$ is the sequence $\{G_t = (V, E_t) : t \in \mathbb{N}\}$ of snapshots such that $E_t = \{e \in E : X_t^e = 1\}$, where $\{X_t^e\}_{t\in\mathbb{N}}$ is a Markov chain for the edge $e \in E$ with states $\{0, 1\}$ (corresponding to non-appearance and appearance of e, respectively) and probability transition matrix:

$$M_e = \left(\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 1-p_e & p_e \\ 1 & q_e & 1-q_e \end{array}\right), \text{ where } 0 \leq p_e, q_e \leq 1.$$

Using this formalism, p_e (resp. q_e) is the probability that the edge e changes its current state from absence to appearance (resp. from appearance to absence) in the next snapshot. Note here that, setting $p_e = p$ and $q_e = q$ for every edge e, we obtain exactly the well-established *edge-Markovian evolving graph* model introduced by Clementi et al. [16].

2.1 The problems

This work studies two main problems, each under the models of stochastic temporal graphs defined above. To describe both of these problems, let us first recall that information in temporal graphs flows via journeys, i.e. temporal paths.

▶ **Definition 3** (Time-edge). *A time-edge in a temporal graph $\mathcal{G} = \{G_t : t \in \mathbb{N}\}$ is a pair (e, t) such that $e \in E_t$.*

▶ **Definition 4** (Journey / temporal path). *Let $\mathcal{G} = \{G_t : t \in \mathbb{N}\}$ be a temporal graph and s, y be two vertices of G. An s-y* journey *(or an s-y* temporal path*) in $\mathcal{G}$ is a sequence $\big((e_1, t_1), \ldots, (e_x, t_x)\big)$ of time-edges over a path $(e_1, \ldots, e_x)$ from s to y in G, where $t_1 < t_2 < \ldots < t_x$. The* arrival time *of the journey is the time t_x of appearance of its last edge.*

▶ **Definition 5** (Foremost Journey). *A foremost s-y journey in a temporal graph $\mathcal{G}$ is an s-y journey with the minimum arrival time amongst all s-y journeys in $\mathcal{G}$.*

Notice that the arrival time of a foremost s-y journey in a stochastic temporal graph is a random variable, which we henceforth denote by $X(s, y)$. The first problem that we study here is how to compute the expected value of the latter, namely $\mathbb{E}[X(s, y)]$.

▷ Problem 1 (Minimum Arrival). Given a stochastic temporal graph on an underlying graph $G = (V, E)$ and two distinct vertices $s, y \in V$, compute the expected value of the arrival time of a foremost s-y journey, i.e. $\mathbb{E}[X(s, y)]$.

Now suppose that an individual (say Alice) is at day 0 at vertex s and would like to arrive at vertex y through a temporal path as quickly as possible. Denote by s_t the vertex where she is located at time t; then $s_0 = s$. Every day t Alice "wakes up" in the morning and looks at which edges are available in today's snapshot; by only knowing her current position, the history of the last k snapshots, and the input parameters of the stochastic temporal graph (i.e. the probabilistic rules of edge appearance), Alice needs to decide whether:

(a) to stay at the vertex s_t she currently is, or

(b) to use an edge of G_t to move to a neighboring vertex.

That is, s_{t+1} is either equal to s_t or equal to some vertex of $\Gamma_{G_t}(s_t)$.

A natural problem we can study here is to compute the expected arrival time of an s-y journey that Alice can follow, using a *best policy*[1] possible, i.e. a policy (sequence of actions) that minimizes her expected arrival time at y. Notice that the arrival time of the journey suggested to Alice by the best policy is a random variable $Y(s, y)$, whose distribution depends on the specific stochastic temporal graph. In particular, in the memoryless model, the expectation of $Y(s, y)$ depends only on the edges' probabilities of appearance. In the memory-k model, the expectation of $Y(s, y)$ also depends on the initial snapshots $G_{-k+1}, \ldots, G_{-1}, G_0$.

▷ Problem 2 (Best Policy). Given a stochastic temporal graph $\mathcal{G}^{(k)}$ on an underlying graph $G = (V, E)$ and two distinct vertices $s, y \in V$, compute $\mathbb{E}_{\mathcal{G}^{(k)}}[Y(s, y)]$.

In particular, we will write $h(s, y) \overset{\text{def}}{=} \mathbb{E}_{\mathcal{G}^{(0)}}[Y(s, y)]$ and $h(s, y, G_0) \overset{\text{def}}{=} \mathbb{E}_{\mathcal{G}^{(1)}}[Y(s, y)]$.

[1] We use the term "policy" here (instead of "strategy") since, as we will see later, this problem can be formulated using a Markov Decision Process (MDP).

Difference between the two problems.

Before we proceed further, we first give an example illustrating that the problems Minimum Arrival and Best Policy are different. In fact, the gap between the solution to Minimum Arrival and the solution to Best Policy can be arbitrarily large: Consider the graph consisting of vertices s and y and $n-2$ vertex disjoint paths of length 2 between s and y. Assume also that, under the memoryless model, every edge incident to s appears each day with probability 1 and every edge incident to y appears each day independently with probability $n^{-0.9}$. Similarly to the above example of the graph with $n-2$ vertex disjoint paths of length 2, here the expected arrival time of a best policy for Alice is $h(s,y) = 1 + n^{0.9}$. On the other hand, the arrival time of the foremost journey from s to y will be equal to the first day after day 1 on which some edge incident to y appears. But the time needed for the latter to happen follows the geometric distribution with success probability $1-(1-n^{-0.9})^{n-2} = 1-o(1)$. Therefore, the expected arrival time of the foremost journey will be $\mathbb{E}[X(s,y)] = 2 + o(1)$, i.e. much smaller than $h(s,y) = 1 + n^{0.9}$.

As a final note, the expected arrival time $\mathbb{E}[X(s,y)]$ of the foremost s-y journey is always upper-bounded by the minimum among the expected values of the arrival times of all s-y journeys in the temporal graph. This is actually implied by a more general and well-known lemma in Probability Theory (Fatou's lemma [17, p. 29]) which establishes that the expected value of the minimum among n random variables is upper-bounded by the minimum among all the variables' expectations.

3 Computing the expected minimum arrival time

3.1 Hardness of exact computation in the memoryless model

In this section we show that, even in the memoryless model, Minimum Arrival is #P-hard in both undirected graphs and directed acyclic graphs (DAGs). In the proof of the following theorem, the edges can be treated either as oriented, in which case we obtain the result for DAGs, or as non-oriented, in which case we obtain the result for undirected graphs.

▶ **Theorem 6.** Minimum Arrival *in the memoryless model is #P-hard.*

▶ **Corollary 7.** *For every $k \geq 0$,* Minimum Arrival *in the memory-k model is #P-hard.*

3.2 The FPTAS for the memoryless model on series-parallel graphs

3.2.1 The case of paths

In this section we will consider a stochastic temporal graph $\mathcal{P}^{(0)} = (P = (V,E), \{p_e\})$ with the underlying graph being a path $P = (s = v_0, v_2, \ldots, v_n = y)$.

▶ **Lemma 8.** $\mathbb{E}[X_{\mathcal{P}^{(0)}}(s,y)] = \sum_{e\in E} \frac{1}{p_e}$.

Let us denote by μ the expectation $\mu \stackrel{\text{def}}{=} \mathbb{E}[X_{\mathcal{P}^{(0)}}(s,y)] = \sum_{e\in E} \frac{1}{p_e}$. Note that

$$\mu = \sum_{i=1}^{\infty} \Pr[X_{\mathcal{P}^{(0)}}(s,y) \geq i]. \tag{1}$$

In the remainder of this section we will show that the first $O(\mu \ln \mu)$ terms of sum (1) already give a very good approximation of μ. In our analysis we will use the following bound.

▶ **Theorem 9** ([26]). *Let $X = \sum_{i=1}^{n} X_i$, where $n \geq 1$ and X_i, $i = 1, \ldots, n$, are independent geometric random variables with parameters $p_1, p_2, \ldots, p_n \in (0, 1]$, respectively. Let $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} \frac{1}{p_i}$. Then for any $\lambda \geq 1$, $\Pr[X \geq \lambda\mu] \leq e^{1-\lambda}$.*

▶ **Lemma 10.** *Let ε be a number such that $0 < \varepsilon \leq 1$. Then*

$$\mu - \sum_{i=1}^{\tau} \Pr[X_{\mathcal{P}^{(0)}}(s, y) \geq i] = \sum_{i=\tau+1}^{\infty} \Pr[X_{\mathcal{P}^{(0)}}(s, y) \geq i] < \varepsilon,$$

for every $\tau \geq \mu \left(\ln \frac{\mu}{\varepsilon} + 1\right)$, where $\mu = \mathbb{E}[X_{\mathcal{P}^{(0)}}(s, y)]$.

3.2.2 A general FPTAS approach

While deriving analytically and computing efficiently the exact solution of Minimum Arrival in a path is an easy task (cf. Lemma 8), it does not seem to be trivial for a slight generalization of paths, called *parallel compositions of paths*. A parallel composition of paths is the graph obtained from a collection of disjoint paths $P_1, P_2, \ldots, P_\ell$ with end vertices s_i, y_i, $i = 1, \ldots, \ell$, respectively, by identifying the vertices $s_1, s_2, \ldots, s_\ell$ in a single vertex s, and by identifying the vertices $y_1, y_2, \ldots, y_\ell$ in a single vertex y.

It is not clear whether there exists an efficient procedure for computing the expected arrival time from s to y in a parallel composition of paths, even if the parallel paths are of equal length and all the probabilities of edge appearance are the same. In this section we present a general approach for developing *ε-additive approximation algorithms*[2] for computing the expected arrival time of a foremost journey in special classes of stochastic temporal graphs. In Section 3.2.3 we apply this approach to develop an efficient ε-additive approximation algorithm for the problem on the class of stochastic temporal graphs with underlying graphs being series-parallel graphs, which generalize parallel compositions of paths and graphs in which all simple s-y paths are of the same length.

Throughout the section we denote by $\mathcal{G}^{(0)} = (G = (V, E), \{p_e\})$ a memoryless stochastic temporal graph with n vertices and m edges, and by $s, y \in V$ two distinct vertices in G. Furthermore, we denote by $H = (V, E, w)$ the weighted graph obtained from the underlying graph G by assigning to every edge $e \in E$ the weight $w(e) = \frac{1}{p_e}$.

▶ **Theorem 11.** *Let $c \in \mathbb{N}$ and $\varepsilon \in (0, 1]$. Let $p_e \geq \frac{1}{n^c}$ for every $e \in E$ and suppose that there exists an algorithm A that computes in time $O(f(\ell, n, m))$ the probabilities $\Pr[X_{\mathcal{G}^{(0)}}(s, y) \geq i]$, for all $i = 1, \ldots, \ell$. Then there exists an algorithm B that approximates $\mathbb{E}[X_{\mathcal{G}^{(0)}}(s, y)]$ within the additive factor of ε in time*

$$O\Big(f\left(n^{c+1} \ln \frac{n}{\varepsilon}, n, m\right) + n \ln n + m\Big).$$

Consequently, if $f(\ell, n, m)$ is a polynomial in variables ℓ, n, and m, then B is an FPTAS on the instance $(\mathcal{G}^{(0)}, s, y)$.

Proof. Let $P = (s = v_0, v_1, \ldots, v_r = y)$ be a minimum weight s-y path in H, and let $\mathcal{P}^{(0)}$ be the stochastic temporal subgraph of $\mathcal{G}^{(0)}$ restricted to the edges of P. For convenience, let us denote $e_i = v_{i-1}v_i$ for every $i = 1, \ldots, r$. Then, by definition and Lemma 8, the

[2] A feasible solution is *ε-additive approximate* if it is within ε additive factor from the optimal value. An algorithm is called an *ε-additive approximation algorithm* if it returns an ε-additive approximate solution for any instance.

weight w^* of P is equal to $\sum_{i=1}^{r} \frac{1}{p_{e_i}} = \mathbb{E}[X_{\mathcal{P}^{(0)}}(s,y)]$. Let $\tau := w^* \left(\ln \frac{w^*}{\epsilon} + 1\right)$. Then, by Lemma 10, we have that

$$\sum_{i=\tau+1}^{\infty} \Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i] \leq \sum_{i=\tau+1}^{\infty} \Pr[X_{\mathcal{P}^{(0)}}(s,y) \geq i] < \varepsilon,$$

and hence

$$\begin{aligned}\sum_{i=1}^{\tau} \Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i] \quad &\leq \quad \mathbb{E}[X_{\mathcal{G}^{(0)}}(s,y)] \quad = \quad \sum_{i=1}^{\infty} \Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i] \\ &< \quad \sum_{i=1}^{\tau} \Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i] + \varepsilon,\end{aligned}$$

that is, $\sum_{i=1}^{\tau} \Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i]$ approximates $\mathbb{E}[X_{\mathcal{G}^{(0)}}(s,y)]$ within the additive factor of ε.

Now we define the desired algorithm B as follows:

1. Construct the graph H and compute the minimum weight w^* of an s-y path in H using Dijkstra's algorithm.
2. Using algorithm A, compute the probabilities $\Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i], i = 1, \ldots, \tau$, where $\tau = w^* \left(\ln \frac{w^*}{\epsilon} + 1\right)$.
3. Output $\sum_{i=1}^{\tau} \Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i]$.

The above discussion implies that algorithm B correctly computes the declared approximation of $\mathbb{E}[X_{\mathcal{G}^{(0)}}(s,y)]$. It remains to justify the time complexity. First, Dijkstra's algorithm can be implemented to work in time $O(n \ln n + m)$ [22]. Second, the assumption on p_e's implies that $w^* = O(n^{c+1})$, and hence $\tau = w^* \left(\ln \frac{w^*}{\epsilon} + 1\right) = O\left(n^{c+1} \ln \frac{n}{\epsilon}\right)$. Therefore the assumption of the theorem implies that the last two steps of the algorithm run in time $O\Big(f\left(n^{c+1} \ln \frac{n}{\varepsilon}, n, m\right)\Big)$, which in turn implies the complexity bound and completes the proof. ◀

3.2.3 The FPTAS for stochastic temporal series-parallel graphs

In the present section we use the approach from Section 3.2.2 to derive a polynomial-time approximation scheme for stochastic temporal series-parallel graphs.

▶ **Theorem 12.** *Let $\varepsilon \in (0,1]$ and let $\mathcal{G}^{(0)} = \{G = (V,E), \{p_e\}\}$ be a stochastic temporal series-parallel graph, where s and y are the terminals of G and $p_e \geq \frac{1}{n^c}$ for every $e \in E$. Then* Minimum Arrival *on $\mathcal{G}^{(0)}$ admits an FPTAS with running time $O\Big(m \cdot n^{2c+2} \ln^2 \frac{n}{\varepsilon}\Big)$, where $|V| = n$ and $|E| = m$.*

3.3 The FPRAS for general graphs in the memory-k model, $k \geq 0$

In this section, we present our FPRAS for Minimum Arrival in the memory-k model, for every $k \geq 0$, under the assumption that the appearance probability of every edge e is lower bounded by $\frac{1}{n^c}$ for some $c \geq 1$ regardless of the history $H_e^{(k)}$, i.e. $p_e(x) \geq \frac{1}{n^c}$ holds for all $x \in \{0,1\}^k$.

▶ **Theorem 13.** *Let $\varepsilon \in (0,1)$ and let $\mathcal{G}^{(k)}$ be a memory-k stochastic temporal graph with two designated vertices s, y. Furthermore let every edge appearance probability be at least $\frac{1}{n^c}$ for some $c \geq 1$, regardless of the history $H_e^{(k)}$ of e. Then* Minimum Arrival *admits an FPRAS which runs in $O\left(m \frac{n^{5c+8}}{\varepsilon^4} \cdot \log(\frac{n}{\varepsilon})\right)$ time with probability of success at least $1 - \frac{2}{n}$.*

4 Computing the expected arrival time of a best policy

In this section we investigate the computational complexity of our second problem, namely Best Policy.

4.1 A polynomial-time algorithm for the memoryless model

In this section we focus on the memoryless model and we derive a polynomial-time dynamic-programming algorithm for Best Policy. We define for every vertex v the expected arrival time $h(v, y) \stackrel{\text{def}}{=} \mathbb{E}_{\mathcal{G}^{(0)}}[Y(v, y)]$ of the v-y journey suggested to Alice by a best policy (i.e. when Alice starts her journey at vertex v). For simplicity of presentation, throughout Section 4.1 we write $h(v) \stackrel{\text{def}}{=} h(v, y)$.

Assume for now that for all $v \in V$, the value $h(v)$ is given; let $v_1 = y, v_2, \dots, v_n$ be an ordering of vertices of V in non-decreasing values of h (ties broken arbitrarily), namely $h(v_1) \leq h(v_2) \leq \dots \leq h(v_n)$. Clearly, $v_1 = y$ and $h(v_1) = h(y) = 0$.

Let s_t be the vertex that Alice occupied at time t and recall that $\Gamma_{G_t}(v)$ is the neighborhood of vertex v in the snapshot G_t, for all $v \in V$ and all $t \in \mathbb{N}$. Notice that, the best strategy of Alice at time $t+1$ is to look at all neighboring vertices of s_t in G_{t+1} and find one with minimum h-value, namely a vertex $u \in \arg\min\{h(v) : v \in \Gamma_{G_{t+1}}(s_t)\}$. If $h(u) \geq h(s_t)$, then Alice has no incentive to change vertex and thus $s_{t+1} = s_t$. Otherwise, if $h(u) < h(s_t)$, then $s_{t+1} = u$.

Therefore, to find the best choice for Alice, it suffices to find the values $h(v), v \in V$. In view of the above, if Alice is on vertex v_i at time 0 (i.e. she is on the i-th best vertex in terms of closeness to y), she will move to the j-th best (with $j < i$) only if an edge appears between v_i and v_j in the next step, and no edge to a vertex better than v_j appears (i.e. no edge between v_i and v_ℓ, $1 \leq \ell \leq j-1$). This happens with probability $Q_{i,j} = p_{\{v_i,v_j\}} \prod_{\ell=1}^{j-1}(1 - p_{\{v_i,v_\ell\}})$, where $\{v_i, v_\ell\}$ denotes the (undirected) edge between v_i and v_ℓ. Additionally, with probability $Q_i = \prod_{\ell=1}^{i-1}(1 - p_{\{v_i,v_\ell\}})$ no edge to a vertex better than v_i will appear, in which case Alice will stay on v_i. Therefore $h(v_i)$ can be recursively computed by $h(v_i) = \sum_{j=1}^{i-1} Q_{i,j} h(v_j) + Q_i h(v_i) + 1$, or equivalently $h(v_i) = \dfrac{\sum_{j=1}^{i-1} Q_{i,j} h(v_j) + 1}{1 - Q_i}$, with initial condition $h(v_1) = 0$. Indeed, the above equation follows by observing that the expected length of the foremost journey to y when Alice is on v_i is equal to $1 + h(v_1)$ with probability $Q_{i,1}$ (which is the probability that an edge between v_i and $v_1 = y$ exists), plus $1 + h(v_2)$ with probability $Q_{i,2}$ (which is the probability that an edge between v_i and the second best vertex v_2 exists, but there is no edge between v_i and v_1), and so on. In general, the above recurrence states that there is no incentive to visit vertices with larger index and also Alice will visit the smallest index vertex v_j for which the edge $\{v_i, v_j\}$ is present (otherwise, if no such edge exists, she will stay on v_i). Using the above recurrence, we can compute all values of $h(v_i)$ by a bottom-up dynamic programming algorithm.

▶ **Theorem 14.** Best Policy *can be optimally computed in the memoryless model in* $O(n^2)$ *time and space.*

4.2 Hardness of computation for the memory-k model, $k \geq 3$

We now show that Best Policy is #P-hard for memory-3 stochastic temporal graphs on directed acyclic graphs, and consequently also for memory $k \geq 3$.

▶ **Theorem 15.** *When the underlying graph is a Directed Acyclic Graph (DAG), it is #P-hard to compute the expected arrival time of the best policy journey in the memory-3 model.*

Proof. We will provide a reduction from the counting problem #PP2DNF which is known to be #P-hard [35]. This problem takes as input a DNF formula $\Phi = \bigvee_{(i,j)\in E} x_i y_j$ on the sets of variables $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$, for some $E \subseteq [n] \times [m]$, and the task is to compute the number ψ of truth assignments that satisfy Φ. We create a directed acyclic graph (DAG) H as follows. First, H has one vertex for each of the variables in $X \cup Y$; then we add two distinct vertices s, y and one other vertex v. For every vertex $x_i \in X$ and every vertex $y_i \in Y$ we add the directed edges (s, x_i) and (y_j, y). Furthermore we add the edge (x_i, y_j) whenever $x_i y_j$ is a clause in Φ. Finally we add the edges (s, v) and (v, y). The construction of H is illustrated in Figure 1.

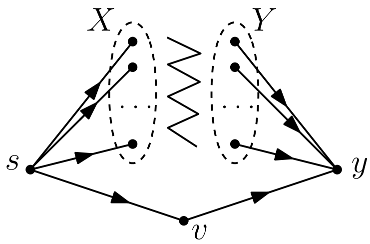

Figure 1 The construction of the DAG H.

Denote by $M = 5 \cdot 2^{n+m}$, and assume that $2^{n+m} \geq 3$ in order to avoid trivialities. All edges (x_i, y_j) appear constantly in H, i.e. they appear at every time step $i \geq 1$ in a memoryless fashion with probability 1. Both edges (s, v) and (v, y) also appear in a memoryless fashion, each of them with probability $\frac{2}{M}$ at every step $i \geq 1$. Moreover, each of the edges (s, x_i) and (y_j, y) appears at each step $i \geq 1$ according to the following table of memory 3. This table has four columns and eight rows. Each column is labeled with the sequence of consecutive time steps $i-3, i-2, i-1$, and i. Each row corresponds to a different triple of appearances of each of the edges in $\{(s, x_i), (y_j, y) : x \in X, y \in Y\}$ at the time steps $i-3, i-2, i-1$ (here 1 means "edge exists" and 0 means "edge does not exist"). At the end of each row there is a pair of numbers $(p, 1-p)$ which denotes that, with the particular history of memory 3, at time step i the edge appears with probability p and it does not appear with probability $1-p$. For simplicity of notation, in the column of time step i, we write "0" and "1" to denote the entries $(0, 1)$ and $(1, 0)$, respectively.

$i-3$	$i-2$	$i-1$	i
0	0	1	0
0	1	0	$(\frac{1}{2}, \frac{1}{2})$
1	0	0	0
0	0	0	0
1	0	1	1
0	1	1	1
1	1	1	1
1	1	0	1

To complete the description of our memory-3 instance, we specify that, in the fictitious initialization snapshots G_{-2}, G_{-1}, G_0, each of the edges (s, x_i) and (y_j, y) appears with probability 0, 0, and 1, respectively, i.e. according to the first row of the above table.

The intuition of this table for the edges (s, x_i) and (y_j, y) is as follows. In the snapshot G_1, none of these edges appears (see the first line of the table). Then, to determine whether each of these edges appears at time step 2 (see the second row of the table), we need to toss an unbiased coin which with probability $\frac{1}{2}$ outputs "appear" and with probability $\frac{1}{2}$ outputs "does not appear". Once this coin has been tossed at time step 2, the status of the edge does not change any more in any subsequent time step $i \geq 3$. That is, if one of the edges (s, x_i) and (y_j, y) appears (resp. does not appear) at time 2, then it appears (resp. does not appear) at all times $i \geq 3$ too. This is easy to be verified by observing the rows 3-7 of the table. Note that the last row of the table is included only for the sake of completeness, as it does not affect the appearance of any edge of H at any time step i.

Let ℓ be the expected s-y arrival time of the best policy in the memory-3 model. Note that, from the above construction of the temporal graph instance, each of the edges (s, x_i) and (y_j, y) appears with probability $\frac{1}{2}$ at all steps $i \geq 2$, while it does not appear at any step $i \geq 2$ with probability $\frac{1}{2}$. Therefore, the probability that there exists a directed temporal path (s, x_i, y_j, y) is equal to $g = \frac{\psi}{2^{n+m}}$, where ψ is the number of satisfying truth assignments of the DNF formula Φ. That is, with probability $1 - g$, there exists no such temporal path from s to y with 3 edges through some vertices x_i and y_j. Furthermore, the expected s-y arrival time through the edges (s, v) and (v, y) is equal to $\frac{M}{2} + \frac{M}{2} = M$. Therefore, since with probability $1 - g$ any policy (also the best one) needs to travel from s to y through vertex v, it follows that $\ell \geq M(1 - g)$.

We now define the following policy: at time step 1 do nothing and just wait for the outcome of the random coin tosses which occur at time step 2. Subsequently, at time step 2 do the following: if there exists a directed temporal path (s, x_i, y_j, y) then follow it, starting at time step 2; otherwise follow the temporal path (s, v, y) which has an expected travel time $\frac{M}{2} + \frac{M}{2} = M$. The expected arrival time of this particular policy is equal to $1 + 3g + M(1 - g)$, and thus it follows that $\ell \leq 1 + 3g + M(1 - g)$. Summarizing, we have:

$$
\begin{aligned}
M(1-g) \quad &\leq \quad \ell \quad \leq \quad 1 + 3g + M(1-g) \Leftrightarrow \\
5 \cdot 2^{n+m} - 5\psi \quad &\leq \quad \ell \quad \leq \quad 5 \cdot 2^{n+m} - 5\psi + 3\frac{\psi}{2^{n+m}} + 1.
\end{aligned}
$$

The first inequality can be written as $2^{n+m} - \frac{\ell}{5} \leq \psi$, while the second one can be written as $\left(1 - \frac{3}{5 \cdot 2^{n+m}}\right)\psi \leq 2^{n+m} - \frac{\ell}{5} + \frac{1}{5}$. Therefore:

$$
2^{n+m} - \frac{\ell}{5} \leq \psi \leq \left(1 + \frac{3}{5 \cdot 2^{n+m} - 3}\right)\left(2^{n+m} - \frac{\ell}{5} + \frac{1}{5}\right) \leq 2^{n+m} - \frac{\ell}{5} + \frac{1}{5} + \frac{3}{4},
$$

and thus $2^{n+m} - \frac{\ell}{5} \leq \psi \leq 0.95 + 2^{n+m} - \frac{\ell}{5}$. Therefore, knowing the expected value ℓ for the best policy we can derive the exact integer value for ψ in the counting problem #PP2DNF. This completes the #P-hardness reduction. ◀

4.3 An exact algorithm for the memory-k model, $k \geq 1$

In this section we present a doubly exponential-time exact algorithm for computing the best policy for Alice in the memory-k model, where $k \geq 1$. Our results in this section are derived using a Markov Decision Process (MDP) formulation of our problem under the memory-k model.

▶ **Theorem 16.** *Let $k \geq 1$ and $\mathcal{G}^{(k)}$ be a stochastic temporal graph, where the underlying graph G has n vertices and m edges. Then* Best Policy *can be solved on $\mathcal{G}^{(k)}$ in $O(2^{(kmn+n\log n)\cdot 2^{km}})$ time.*

References

1 E. Aaron, D. Krizanc, and E. Meyerson. DMVP: foremost waypoint coverage of time-varying graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 29–41, 2014.

2 E. Akrida, G.B. Mertzios, S. Nikoletseas, C. Raptopoulos, P.G. Spirakis, and V. Zamaraev. *How fast can we reach a target vertex in stochastic temporal graphs?*, 2019. Technical Report available at `https://arxiv.org/abs/1903.03636`.

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

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

5 E.C. Akrida, G.B. Mertzios, P.G. Spirakis, and V. Zamaraev. Temporal vertex cover with a sliding time window. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 148:1–148:14, 2018.

6 A. Anagnostopoulos, J. Lacki, S. Lattanzi, S. Leonardi, and M. Mahdian. Community detection on evolving graphs. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, pages 3522–3530, 2016.

7 C. Avin, M. Koucký, and Z. Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *International Colloquium on Automata, Languages and Programming, ICALP*, pages 121–132, 2008.

8 P. Basu, A. Bar-Noy, R. Ramanathan, and M. P. Johnson. Modeling and analysis of time-varying graphs. *CoRR*, abs/1012.0260, 2010.

9 P. Basu, S. Guha, A. Swami, and D. Towsley. Percolation phenomena in networks under random dynamics. In *International Conference on Communication Systems and Networks, COMSNETS*, pages 1–10, 2012.

10 P. Basu, F. Yu, A. Bar-Noy, and D. Rawitz. To sample or to smash? estimating reachability in large time-varying graphs. In *SIAM International Conference on Data Mining*, pages 983–991, 2014.

11 P. Basu, F. Yu, M. P. Johnson, and A. Bar-Noy. Low expected latency routing in dynamic networks. In *IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS*, pages 267–271, 2014.

12 A. Casteigts and P. Flocchini. Deterministic Algorithms in Dynamic Networks: Formal Models and Metrics. Technical report, Defence R&D Canada, April 2013. URL: `https://hal.archives-ouvertes.fr/hal-00865762`.

13 A. Casteigts and P. Flocchini. Deterministic Algorithms in Dynamic Networks: Problems, Analysis, and Algorithmic Tools. Technical report, Defence R&D Canada, April 2013. URL: `https://hal.archives-ouvertes.fr/hal-00865764`.

14 A. Casteigts, P. Flocchini, E. Godard, N. Santoro, and M. Yamashita. On the expressivity of time-varying graphs. *Theoretical Computer Science*, 590:27–37, 2015.

15 A. Clementi, A. Monti, F. Pasquale, and R. Silvestri. Information spreading in stationary markovian evolving graphs. *IEEE Transactions on Parallel and Distributed Systems*, 22(9):1425–1432, 2011.

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

17 R. Durrett. Probability: Theory and examples, 2011.

18 J. Enright, K. Meeks, G.B. Mertzios, and V. Zamaraev. *Deleting edges to restrict the size of an epidemic in temporal networks*, 2018. Technical Report available at `https://arxiv.org/abs/1805.06836`.

19 T. Erlebach, M. Hoffmann, and F. Kammer. On temporal graph exploration. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 444–455, 2015.

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

21 P. Flocchini, B. Mans, and N. Santoro. On the exploration of time-varying networks. *Theoretical Computer Science*, 469:53–68, 2013.

22 M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, 1987.

23 S. Henri, S. Shneer, and P. Thiran. On the delays in time-varying networks: Does larger service-rate variance imply larger delays? In *ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc*, pages 201–210, 2018.

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

25 P. Holme and J. Saramäki, editors. *Temporal Networks*. Springer, 2013.

26 S. Janson. Tail bounds for sums of geometric and exponential variables. *Statistics & Probability Letters*, 135:1–6, 2018.

27 D. Kempe, J. M. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. In *ACM Symp. on Theory of Comp. (STOC)*, pages 504–513, 2000.

28 I. Lamprou, R. Martin, and P. G. Spirakis. Cover time in edge-uniform stochastically-evolving graphs. *Algorithms*, 11(10):149, 2018.

29 G.B. Mertzios, O. Michail, I. Chatzigiannakis, and P.G. Spirakis. Temporal network optimization subject to connectivity constraints. In *International Colloquium on Automata, Languages and Programming (ICALP), Part II*, pages 657–668, 2013.

30 G.B. Mertzios, H. Molter, and V. Zamaraev. Sliding window temporal graph coloring. In *Proceedings of the 33st AAAI Conference on Artificial Intelligence (AAAI)*, 2019. To appear.

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

32 P. Nain, D. Towsley, M. P. Johnson, P. Basu, A. Bar-Noy, and F. Yu. *Computing Traversal Times on Dynamic Markovian Paths*, 2013. Technical Report available at `http://arxiv.org/abs/1303.3660`.

33 A. Orda and R. Rom. Distributed shortest-path protocols for time-dependent networks. *Distributed Computing*, 10(1):49–62, 1996.

34 B. Pittel. On spreading a rumor. *SIAM Journal on Applied Mathematics*, 47(1):213–223, 1987.

35 J.S. Provan and M.O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4):777–788, 1983.

36 N. Santoro. Computing in time-varying networks. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems SSS*, page 4, 2011.

37 C. Scheideler. Models and techniques for communication in dynamic networks. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 27–49, 2002.

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