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

Eleni C. Akrida* George B. Mertzios† Sotiris Nikoletseas‡
Christoforos Raptopoulos§ Paul G. Spirakis¶ Viktor Zamaraev‖

Abstract

Temporal graphs abstractly model real-life inherently dynamic networks. Given a graph G, a temporal graph with G as the underlying graph is a sequence of subgraphs (snapshots) G_t of G, where $t \geq 1$. In this paper we study stochastic temporal graphs, i.e. stochastic processes $\mathcal{G}$ whose random variables are the snapshots of a temporal graph on G. A natural feature observed in various real-life scenarios is a memory effect in the appearance probabilities of particular edges; i.e. the probability an edge $e \in E$ appears at time step t depends on its appearance (or absence) at the previous k steps. We study the hierarchy of models of memory-k, $k \geq 0$, in an edge-centric network evolution setting: every edge of G has its own independent probability distribution for its appearance over time. We thoroughly investigate the complexity of two naturally related, but fundamentally different, temporal path problems, called Minimum Arrival and Best Policy.

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

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, 38]. 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* [5, 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

*Department of Computer Science, University of Liverpool, Liverpool, UK. Email: e.akrida@liverpool.ac.uk
†Department of Computer Science, Durham University, Durham, UK. Email: george.mertzios@durham.ac.uk
‡Computer Engineering & Informatics Department, University of Patras and CTI, Greece. Email: nikole@cti.gr
§Computer Engineering & Informatics Department, University of Patras and CTI, Greece. Email: raptopox@ceid.upatras.gr
¶Department of Computer Science, University of Liverpool, UK and Computer Engineering & Informatics Department, University of Patras, Greece. Email: p.spirakis@liverpool.ac.uk
‖Department of Computer Science, Durham University, Durham, UK. Email: viktor.zamaraev@durham.ac.uk

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, 6, 7, 9, 14, 18, 19, 21, 23, 28, 35]. In addition to temporal paths, recently also various temporal non-path problems have been introduced and algorithmically studied, such as temporal vertex cover [2], temporal coloring [30], and temporal Δ-cliques [24, 42].

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 [16, 36, 39]. 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. These models concern 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. [15], 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 [15]. It is worth noting here that Avin et al. [6] 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 [6]. 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. [8] 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. [7] 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. Ogier and Rutenburg [34] consider the memory-1 case and study a slightly different (and harder) problem in which they also allow arbitrary link delays. They show that their problem is in general #P-hard, by giving a reduction from the two-terminal reliability problem, and provide some polynomial algorithms for special cases. Special cases of the memory-1 model were also considered in [10].

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 [37].

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 3) 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 $c \geq 1$ regardless of the history of the edge. 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 4.3.2 an exact doubly exponential-time algorithm with running time $O(2^{(kmn+n\log n)\cdot 2^{km}})$.

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 $H = (V_H, E_H)$ of G, denoted by $H \subseteq G$, is a graph where $V_H \subseteq V$ and $E_H \subseteq E$. A spanning subgraph $H = (V, E_H)$ of G, denoted by $H \subseteq_s G$, is a graph on vertex set V 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 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. [15].

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:

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

(ii) 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. In this context, a *policy* is a function $\pi : V \times 2^G \to V$ that maps a pair $u \in V$, $G' \subseteq_s G$ of Alice's current vertex $u \in V$ and the current snapshot G' to a vertex $v \in \Gamma_{G'}[u]$, where 2^G denotes the set of all spanning subgraphs of G. Vertex $v = \pi(u, G')$ is then the new (current) location of Alice in her journey towards 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 $h(s, y) \stackrel{\text{def}}{=} \mathbb{E}_{\mathcal{G}^{(0)}}[Y(s, y)]$ and $h(s, y, G_0) \stackrel{\text{def}}{=} \mathbb{E}_{\mathcal{G}^{(1)}}[Y(s, y)]$.

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. To demonstrate this, assume the memoryless model $\mathcal{G}^{(0)}$ and consider the 4-cycle a, b, c, d, a as the underlying graph. Let $s = a$ and $y = c$ and assume that, at any time step, each edge appears independently with probability $\frac{1}{2}$.

Any best policy for Alice will wait until an edge incident to a appears and then cross it; if both adjacent edges (a, b) and (a, d) appear at the same time, then it does not matter which one she chooses. The event "some edge adjacent to a appears" occurs with probability $\frac{3}{4}$, hence, the expected time until such an edge appears is $\frac{4}{3}$. Furthermore, when Alice reaches one of the vertices b or d, an optimal policy will never suggest going back to a, so Alice will have to wait until the last edge to c appears, which takes 2 steps on expectation. Overall, the optimal policy for Alice will take $h(a, c) = \frac{10}{3}$ steps on expectation. This is the solution to Best Policy (see Problem 2).

On the other hand, Minimum Arrival (see Problem 1) asks for the expectation of the arrival time $X(a, c)$ of a foremost s-y journey. To compute $\mathbb{E}[X(a, c)]$, denote by T_b (resp. T_d) the arrival time of a journey allowed to use only edges (a, b) and (b, c) (resp. (a, d) and (d, c)), when they appear. Then,

$$X(a, c) > k \quad \Leftrightarrow \quad T_b > k \text{ and } T_d > k$$

[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).

But the probability of the event $\{T_b > k\}$ is equal to the probability that either (a, b) does not appear until (and including) step k plus the probability that it appears within the first k steps, and (b, c) does not appear after that until (and including) k. Therefore,

$$\Pr[T_b > k] = \frac{1}{2^k} + k\frac{1}{2}\frac{1}{2^{k-1}} = (k+1)\frac{1}{2^k}.$$

By symmetry we have $\Pr[T_b > k] = \Pr[T_d > k]$ and, by independence, for any $k \geq 2$:

$$\Pr[X(a, c) > k] = \Pr[T_b > k] \Pr[T_d > k] = \frac{1}{2^{2k}} + k\frac{1}{2^{2k-1}} + k^2\frac{1}{2^{2k}}.$$

By using the fact that $\mathbb{E}[X(a, c)] = \sum_{k=0}^{\infty} \Pr[X(a, c) > k] = 2 + \sum_{k=2}^{\infty} \Pr[X(a, c) > k]$, it follows that $\mathbb{E}[X(a, c)] = 2 + \frac{26}{27} = \frac{80}{27}$, which is strictly smaller than $\frac{10}{3}$.

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 $n^{-0.9}$. 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 1. Minimum Arrival *in the memoryless model is #P-hard.*

Proof. To prove the theorem we will provide a reduction from the #P-complete problem #PP2DNF [37]. The latter problem is defined as follows. Let $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_m\}$ be two disjoint sets of Boolean variables. A *positive*, *partitioned* 2-DNF formula is a DNF formula of the form:

$$\Phi = \bigvee_{(i,j)\in E} x_i y_j,$$

for some $E \subseteq [n] \times [m]$. Given a positive, partitioned 2-DNF formula Φ, the problem #PP2DNF asks for the number of truth assignments satisfying Φ. Let Φ be an instance of #PP2DNF. We define G to be a graph with the vertex set $\{s, y\} \cup X \cup Y$ and the edge set $\{(s, x_i) \mid x_i \in X\} \cup \{(x_i, y_j) \mid (i, j) \in E\} \cup \{(y_i, y) \mid y_j \in Y\}$, see Figure 1.

First we claim[2] that the number ψ of satisfying assignments of Φ is equal to the number of spanning subgraphs of G which contain all the edges from $\{(x_i, y_j) \mid (i, j) \in E\}$ and have a simple path from s to y of length 3. To see the claim, for every subset $S \subseteq \{(s, x_i) \mid x_i \in X\} \cup \{(y_i, y) \mid y_j \in Y\}$ of edges we define a truth assignment α that assigns $x_i = 1$ iff $(s, x_i) \in S$ and $y_j = 1$ iff $(y_j, y) \in S$. Notice that every s-y path of length 3 in G is of the form (s, x_i, y_j, y) for some $(i, j) \in E$. Therefore, if the

[2]This claim was provided by Antoine Amarilli (`https://cstheory.stackexchange.com/q/42239`).

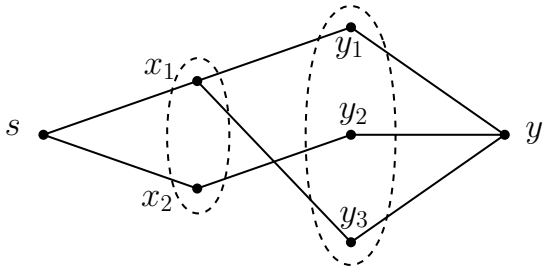

Figure 1: Example construction of G, given the positive, partitioned 2-DNF formula $\Phi = (x_1y_1) \vee (x_1y_3) \vee (x_2y_2)$.

subgraph spanned by S contains a path (s, x_i, y_j, y), then α assigns 1 to both x_i and y_j, and hence α satisfies Φ. Conversely, given an assignment α satisfying Φ, we define a subgraph of G spanned by the edge set $\{(s, x_i) \mid x_i \text{ is assigned 1 by } \alpha\} \cup \{(x_i, y_j) \mid (i, j) \in E\} \cup \{(y_i, y) \mid y_j \text{ is assigned 1 by } \alpha\}$. Since α is satisfying assignment, there exists $(i, j) \in E$ such α assigns 1 to both x_i and y_j, and therefore the subgraph contains the s-y path (s, x_i, y_j, y) of length 3.

Now we define an instance of Minimum Arrival in the memoryless model as follows. Let H be the graph obtained from G by adding three new vertices v_1, v_2, v_3 and four new edges $(s, v_1), (v_1, v_2), (v_2, v_3), (v_3, y)$, which all together form a new s-y path of length 4. For every edge $e \in \{(s, x_i) \mid x_i \in X\} \cup \{(y_i, y) \mid y_j \in Y\}$ we set $p_e = 1/2$, and for any other edge e of H we set $p_e = 1$. In this stochastic temporal graph the duration of a foremost journey from s to y is either 3, if for some $(i, j) \in E$ the edge (s, x_i) appears in time slot 1, and the edge (y_j, y) appears in time slot 3, or 4 otherwise. In other words, the duration of a foremost s-y journey depends only on the subgraph of G spanned by the edge set $R_1 \subseteq \{(s, x_i) \mid x_i \in X\}$ that appears in slot 1, and by the edge set $R_3 \subseteq \{(y_i, y) \mid y_j \in Y\}$ that appears in slot 3. The duration is equal to 3 if and only if the subgraph of G spanned by $R_1 \cup \{(x_i, y_j) \mid (i, j) \in E\} \cup R_3$ has an s-y path of length 3. Since every edge in $R_1 \cup R_3$ appears independently with probability $1/2$, it follows that the probability that this subgraph has a path of length 3 is equal to $p = \frac{\psi}{2^{n+m}}$. Consequently,

$$\mathbb{E}[X(s, y)] = 3p + 4(1 - p) = 4 - p,$$

and hence $\psi = 2^{n+m}(4 - \mathbb{E}[X(s, y)])$. Therefore, knowing the expected duration $\mathbb{E}[X(s, y)]$ of an s-y foremost journey, we can efficiently compute the number of satisfying assignments of Φ, which proves that the computation of $\mathbb{E}[X(s, y)]$ is #P-hard. □

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

Proof. Consider a stochastic temporal graph with a single edge e which appears every day independently with probability p_e, and let X_e be a random variable equal to the duration of the foremost journey from one of the endpoints of e to the other. Then $X_{\mathcal{P}^{(0)}}(s, y) = \sum_{e \in E} X_e$. Notice that X_e is a geometric random variable with probability mass function $\Pr[X_e = i] = (1 - p_e)^{i-1} p_e$ for $i = 1, 2, 3, ...$, and expectation $\mathbb{E}[X_e] = \frac{1}{p_e}$. Therefore $\mathbb{E}[X_{\mathcal{P}^{(0)}}(s, y)] = \sum_{e \in E} \mathbb{E}[X_e] = \sum_{e \in E} \frac{1}{p_e}$. □

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 2 ([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 2. *Let ε be a number such that $0 < \varepsilon \leq 1$. Then*

$$\mu - \sum_{i=1}^{\tau} \Pr[X_{\mathcal{P}^{(0)}}(s, y) \geq i] = \sum_{i=\tau+1}^{\infty} \Pr[X_{\mathcal{P}^{(0)}}(s, y) \geq i] < \varepsilon, \tag{2}$$

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

Proof. The equality in (2) follows from (1). In the rest of the proof we show the inequality. Since $\tau \geq \mu \left(\ln \frac{\mu}{\varepsilon} + 1\right) \geq \mu$, using Theorem 2 we have

$$\begin{aligned}\sum_{i=\tau+1}^{\infty} \Pr[X_{\mathcal{P}^{(0)}}(s, y) \geq i] &\leq \sum_{i=\tau+1}^{\infty} e^{1-\frac{i}{\mu}} = \frac{e^{1-\tau/\mu}}{e^{1/\mu} - 1} \leq \frac{e^{1-\mu\left(\ln \frac{\mu}{\varepsilon}+1\right)/\mu}}{e^{1/\mu} - 1} = \\ &= \frac{\varepsilon}{\mu(e^{1/\mu} - 1)} \leq \frac{\varepsilon}{\mu(1 + \frac{1}{\mu} + \frac{1}{2\mu^2} - 1)} = \frac{\varepsilon}{1 + \frac{1}{2\mu}} < \varepsilon,\end{aligned}$$

where we used the inequality $e^x \geq 1 + x + x^2/2$ which holds for every $x \geq 0$. □

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 1), 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*[3] 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 $w(e) = \frac{1}{p_e}$.

Definition 6. *Let $\mathcal{G}^{(0)}$ be a memoryless stochastic temporal graph, where G is the underlying graph. A* stochastic temporal subgraph *$\mathcal{H}^{(0)}$ of $\mathcal{G}^{(0)}$ is a stochastic temporal graph which has a subgraph $H \subseteq G$ as an underlying graph and inherits all edge appearance probabilities from $\mathcal{G}^{(0)}$.*

Observation 1. *Let $\mathcal{H}^{(0)}$ be a* stochastic temporal subgraph *of the stochastic temporal graph $\mathcal{G}^{(0)}$. Then for every natural number i we have $\Pr[X_{\mathcal{G}^{(0)}}(s, y) \geq i] \leq \Pr[X_{\mathcal{H}^{(0)}}(s, y) \geq i]$, and hence $\mathbb{E}[X_{\mathcal{G}^{(0)}}(s, y)] \leq \mathbb{E}[X_{\mathcal{H}^{(0)}}(s, y)]$.*

The following lemma is a direct consequence of Observation 1 and Lemma 1.

[3] 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.

Lemma 3. *Let w^* be the minimum weight of an s-y path in H. Then $\mathbb{E}[X_{\mathcal{G}^{(0)}}(s,y)] \leq w^*$.*

Theorem 3. *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 $O\left(f(\ell,n,m)\right)$ 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*

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

for some constant d. 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 1, the 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 $\tau := w^* \left(\ln \frac{w^*}{\varepsilon} + 1\right)$. Then, by Observation 1 and Lemma 2, 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

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

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 $\Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i]$ for every $i = 1,\ldots,\tau$, where $\tau = w^* \left(\ln \frac{w^*}{\varepsilon} + 1\right)$.
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 $\tau = w^* \left(\ln \frac{w^*}{\varepsilon} + 1\right) = O\left(n^{c+1} \ln \frac{n}{\varepsilon}\right)$. Therefore the assumption of the theorem implies that the last two steps of the algorithm run in time $O\Big(f\big(d \cdot n^{c+1} \ln \frac{n}{\varepsilon}, n, m\big)\Big)$, for some constant d, 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. According to Theorem 3, the development of such an algorithm reduces to the design of an efficient procedure of computing probabilities of the form $\Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i]$, which is the main goal of this section.

Let G be a graph and s and y be two distinct vertices in G. The triple (G,s,y) is a *two-terminal series-parallel* graph, with terminals s and y, if G can be constructed by a sequence of the following two operations starting from a set of copies of a single-edge two-terminal series-parallel graph (K_2, a, b).

1. *Parallel composition*: Given a pair of two-terminal series-parallel graphs (H_1, s_1, y_1) and (H_2, s_2, y_2), form a new two-terminal series-parallel graph (G,s,y) by identifying $s = s_1 = s_2$ and $y = y_1 = y_2$.

2. *Series composition*: Given a pair of two-terminal series-parallel graphs (H_1, s_1, y_1) and (H_2, s_2, y_2), form a new two-terminal series-parallel graph (G, s, y) by identifying $s = s_1$, $y_1 = s_2$, and $y = y_2$.

Finally, a graph G is called *series-parallel* if (G, s, y) is a two-terminal series-parallel graph for some pair of distinct vertices s and y of G.

The sequence of parallel and series compositions leading to a two-terminal series-parallel graph $(G = (V, E), s, y)$ can be conveniently represented by a decomposition tree. A binary tree $T = (V_T, E_T)$ with a labeling function $\sigma : V_T \to \{\mathtt{s},\mathtt{p}\} \cup E \times \{0, 1\}$ is called a *decomposition tree* of a two-terminal series-parallel graph (G, s, y) if and only if the leaves of T are labeled with elements of $E \times \{0, 1\}$ such that every $e \in E$ appears in exactly one label, internal nodes are labeled with $\mathtt{p}$ or $\mathtt{s}$, and G can be generated recursively using T as follows: If T is a single node v with $\sigma(v) = (e, \alpha)$, then G consists of the single edge e with the source being the vertex with the smallest ID, if $\alpha = 0$, and with the source being the vertex with the largest ID, if $\alpha = 1$. Otherwise, let T_1 (resp. T_2) be the right (resp. left) subtree of T and (H_1, s_1, y_1) and (H_2, s_2, y_2) be two-terminal series-parallel graphs with decomposition trees T_1 and T_2: if $\sigma(v) = \mathtt{p}$ (resp. $\mathtt{s}$) then G is the parallel (resp. series) composition of (H_1, s_1, y_1) and (H_2, s_2, y_2).

We will make use of tree decompositions of series-parallel graphs in our algorithm. It is known that a tree decomposition of a series-parallel graph can be constructed in linear time.

Theorem 4 ([41])**.** *Given a two-terminal series-parallel graph with n vertices and m edges, its tree decomposition can be computed in time $O(n + m)$.*

Let $\mathcal{G}^{(0)} = (G = (V, E), \{p_e\})$ be a stochastic temporal graph with the underlying graph G being series-parallel. Let also $s, y \in V$ be two distinct vertices in G such that (G, s, y) is a two-terminal series-parallel graph. We will present a dynamic programming algorithm which, for a given natural number ℓ, computes the set of probabilities:

$$\Pr[X_{\mathcal{G}^{(0)}}(s, y) \geq i], \quad i = 1, \ldots, \ell. \tag{3}$$

For convenience, the algorithm will also support the set of probabilities:

$$\Pr[X_{\mathcal{G}^{(0)}}(s, y) = i], \quad i = 1, \ldots, \ell - 1. \tag{4}$$

Notice that having computed one of the sets of probabilities, the other set can be computed in $O(\ell^2)$ time.

The algorithm is based on the following recursive equations. Since (G, s, y) is a two-terminal series-parallel graph, it is either a single-edge graph, or can be obtained from smaller two-terminal series-parallel graphs (H_1, s_1, y_1), (H_2, s_2, y_2) via one of the two composition operations.

1. In the case of a single-edge graph we have for every $i \in [\ell - 1]$ that:

$$\Pr[X_{\mathcal{G}^{(0)}}(s, y) = i] = (1 - p)^{i-1} p, \tag{5}$$

   where p is the probability of appearance of the unique edge of the graph.

2. In the case of parallel composition we have for every $i \in [\ell]$ that:

$$\Pr[X_{\mathcal{G}^{(0)}}(s, y) \geq i] = \Pr[X_{\mathcal{H}_1^{(0)}}(s_1, y_1) \geq i] \cdot \Pr[X_{\mathcal{H}_2^{(0)}}(s_2, y_2) \geq i], \tag{6}$$

   where $\mathcal{H}_1^{(0)}$ and $\mathcal{H}_2^{(0)}$ are the stochastic temporal subgraphs of $\mathcal{G}^{(0)}$ restricted to the vertices of H_1 and H_2, respectively.

3. In the case of series composition, we have for every $i \in [\ell - 1]$ that:

$$\Pr[X_{\mathcal{G}^{(0)}}(s, y) = i] = \sum_{j=1}^{i-1} \Pr[X_{\mathcal{H}_1^{(0)}}(s_1, y_1) = j] \cdot \Pr[X_{\mathcal{H}_2^{(0)}}(s_2, y_2) = i - j]. \tag{7}$$

Theorem 5. *Given a stochastic temporal series-parallel graph $\mathcal{G}^{(0)} = (G = (V, E), \{p_e\})$, two vertices $s, y \in V$ such that (G, s, y) is a two-terminal series-parallel graph, and a natural number ℓ, Algorithm 1 correctly computes the probabilities $\{\Pr[X_{\mathcal{G}^{(0)}}(s, y) \geq i] : i \in [\ell]\}$ and $\{\Pr[X_{\mathcal{G}^{(0)}}(s, y) = i] : i \in [\ell - 1]\}$ in time $O(m\ell^2)$, where $m = |E|$.*

Algorithm 1 COMPUTE SP PROBABILITIES

Input: A stochastic temporal graph $\mathcal{G}^{(0)} = (G, \{p_e\})$ on an underlying two-terminal series-parallel graph (G, s, y) with a tree decomposition T_G, and a natural number ℓ.
Output: The sets $\{\Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i] : i \in [\ell]\}$ and $\{\Pr[X_{\mathcal{G}^{(0)}}(s,y) = i] : i \in [\ell-1]\}$

1: **if** G is a single-edge graph with the unique edge e **then**
2: **for** $i = 1$ to $\ell - 1$ **do**
3: $\Pr[X_{\mathcal{G}^{(0)}}(s,y) = i] = (1-p_e)^{i-1}p_e$
4: Compute the set of probabilities $\{\Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i] : i \in [\ell]\}$
5: **else**
6: (G, s, y) is a composition of two series-parallel subgraphs (H_1, s_1, y_1) and (H_2, s_2, y_2)
7: COMPUTE SP PROBABILITIES$(\mathcal{H}_1^{(0)}, s_1, y_1, T_{H_1}, \ell)$
8: COMPUTE SP PROBABILITIES$(\mathcal{H}_2^{(0)}, s_2, y_2, T_{H_2}, \ell)$
9: **if** (G, s, y) is the parallel composition of (H_1, s_1, y_1) and (H_2, s_2, y_2) **then**
10: **for** $i = 1$ to ℓ **do**
11: $\Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i] = \Pr[X_{\mathcal{H}_1^{(0)}}(s_1,y_1) \geq i] \cdot \Pr[X_{\mathcal{H}_2^{(0)}}(s_2,y_2) \geq i]$
12: Compute the set of probabilities $\{\Pr[X_{\mathcal{G}^{(0)}}(s,y) = i] : i \in [\ell-1]\}$
13: **if** (G, s, y) is the series composition of (H_1, s_1, y_1) and (H_2, s_2, y_2) **then**
14: **for** $i = 1$ to $\ell - 1$ **do**
15: $\Pr[X_{\mathcal{G}^{(0)}}(s,y) = i] = \sum_{j=1}^{i-1} \Pr[X_{\mathcal{H}_1^{(0)}}(s_1,y_1) = j] \cdot \Pr[X_{\mathcal{H}_2^{(0)}}(s_2,y_2) = i-j]$
16: Compute the set of probabilities $\{\Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i] : i \in [\ell]\}$
17: **return** $\{\Pr[X_{\mathcal{G}^{(0)}}(s,y) \geq i] : i \in [\ell]\}$ and $\{\Pr[X_{\mathcal{G}^{(0)}}(s,y) = i] : i \in [\ell-1]\}$

Proof. We start with the analysis of the correctness of the algorithm. First, if the underlying graph of the input stochastic temporal graph is a single-edge graph, then the algorithm computes the required sets of probabilities in lines 2-4 using equations (5). Second, if the underlying graph is not a single-edge graph, then, by definition, (G, s, y) is either the parallel or the series composition of two two-terminal series-parallel graphs (H_1, s_1, y_1) and (H_2, s_2, y_2) whose decomposition trees are the subtrees T_{H_1} and T_{H_2} of T_G rooted at the children of the root of T_G. In the case of parallel composition, the algorithm computes the sets of probabilities in lines 10-12 using equations (6). In the case of series composition, the algorithm computes the sets of probabilities in lines 14-16 using equations (7). In both cases, the computation of the probabilities uses only the corresponding sets of probabilities for the stochastic temporal subgraphs $\mathcal{H}_1^{(0)}$ and $\mathcal{H}_2^{(0)}$, which are computed recursively in lines 7 and 8, respectively.

In order to analyze the complexity of Algorithm 1, we observe that for every node of the decomposition tree of the underlying graph the algorithm makes exactly one recursive call. In each of the calls, the algorithm executes either lines 2-4, or lines 10-12, or lines 14-16. It is easy to check that each of these sets of lines performs $O(\ell^2)$ operations of addition or multiplication. Since T_G is a binary tree and has exactly m leaves, in total T_G has $2m-1$ nodes, and therefore the total time complexity of Algorithm 1 is $O(m\ell^2)$. □

Finally we present an FPTAS for the expected arrival time of a foremost s-y journey in a stochastic temporal series-parallel graph.

The following theorem follows from Theorem 3 and Theorem 5.

Theorem 6. *Algorithm 2 is an FPTAS for* MINIMUM ARRIVAL*, i.e. it approximates* $\mathbb{E}[X_{\mathcal{G}^{(0)}}(s,y)]$ *within the additive factor of* ε *in time* $O\left(m \cdot n^{2c+2} \ln^2 \frac{n}{\varepsilon}\right)$*, where* n *and* m *are the number of vertices and the number of edges in the underlying graph, respectively.*

3.3 The FPRAS for general graphs in the memory-k model

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

Algorithm 2 FPTAS for MINIMUM ARRIVAL in stochastic temporal series-parallel graphs

Input: A stochastic temporal series-parallel graph $\mathcal{G}^{(0)} = (G = (V, E), \{p_e\})$ such that $p_e \geq \frac{1}{n^c}$ for every $e \in E$, and a number $\varepsilon \in (0, 1]$.

Output: Number μ such that $\mathbb{E}[X_{\mathcal{G}^{(0)}}(s, y)] - \varepsilon < \mu \leq \mathbb{E}[X_{\mathcal{G}^{(0)}}(s, y)]$

1: Let $H = (V, E, w)$ be the weighted graph obtained from the underlying graph G by assigning to every edge $e \in E$ the weight $w(e) = \frac{1}{p_e}$
2: Compute the minimum weight w^* of an s-y path in H
3: Let $\tau = w^* \left(\ln \frac{w^*}{\varepsilon} + 1 \right)$
4: Compute a tree decomposition T of (G, s, y)
5: COMPUTE SP PROBABILITIES$(\mathcal{G}^{(0)}, s, y, T, \tau)$
6: **return** $\sum_{i=1}^{\tau} \Pr[X_{\mathcal{G}^{(0)}}(s, y) \geq i]$

$c \geq 1$ regardless of the history $H_e^{(k)}$, i.e. $p_e(x) \geq \frac{1}{n^c}$ holds for all $x \in \{0, 1\}^k$.

Lemma 4. *Let $c \geq 1$ be a constant and let $\mathcal{G}^{(k)} = (G = (V, E), \{p_e(H_e^{(k)})\})$ be a memory-k stochastic temporal graph such that $p_e(x) \geq \frac{1}{n^c}$ holds for all $x \in \{0, 1\}^k$ and $e \in E$. Then we have*

1. $\mathbb{E}[X(s, y)] \leq n^{c+1}$.
2. $\sigma^2(X(s, y)) \leq 2n^{2c+2}$.

Proof. For any s-y path P of G, let X_P denote the random variable of the arrival time of a journey following path P. We then have $X(s, y) = \min\{X_P : P \text{ is an } s\text{-}y \text{ path}\}$. Therefore we have $X(s, y) \leq X_{P_1}$ for any fixed path $P_1 := (s = v_0, v_1, v_2, \ldots, v_{|P_1|} = y)$, where $e_1 = \{v_0, v_1\}, e_2 = \{v_1, v_2\}, \ldots, e_{|P_1|} = \{v_{|P_1|-1}, v_{P_1}\}$. Let X_e^t be the random variable equal to the shortest time required to move from one of the end vertices of e to the other starting at time t. Then $X_{P_1} = \sum_{i=1}^{|P_1|} X_{e_i}^{t_i}$, where $t_0 = 0$ and $t_i = t_{i-1} + X_e^{t_{i-1}}$ for every $i = 1, 2, \ldots, |P_1|$. Since $p_e(x) \geq \frac{1}{n^c}$, note that each $X_{e_i}^{t_i}$ is stochastically dominated by a geometric random variable Z_i with success probability $\frac{1}{n^c}$. Therefore X_{P_1} is stochastically dominated by $\sum_{i=1}^{|P_1|} Z_i$, where the Z_i's are mutually independent. Thus $\mathbb{E}[X_{P_1}] \leq \mathbb{E}[\sum_{i=1}^{|P_1|} Z_i] = |P_1| n^c \leq n^{c+1}$, which proves part (1) of the lemma.

For part (2) of the lemma, by definition of the variance, we have

$$\begin{aligned}
\sigma^2(X(s, y)) &= \mathbb{E}[X^2(s, y)] - \mathbb{E}^2[X(s, y)] \leq \mathbb{E}[X^2(s, y)] && (8)\\
&\leq \mathbb{E}\left[\left(\sum_{i=1}^{|P_1|} Z_i \right)^2 \right] && (9)\\
&= \sum_{i,j} \mathbb{E}[Z_i Z_j] && (10)\\
&\leq n^2 \mathbb{E}[Z_1^2] = n^2 \frac{2 - \frac{1}{n^c}}{\frac{1}{n^{2c}}} \leq 2n^{2c+2}. && (11)
\end{aligned}$$

In particular, the second inequality follows by the stochastic domination argument from part (1) and the fact that $X(s, y)$ is non-negative. The third inequality follows from the independence of Z_i, from the fact that the Z_i's have the same distribution, and from the fact that $\mathbb{E}[Z_i^2] \geq \mathbb{E}[Z_i Z_j]$, for any i, j. □

In the following theorem we provide our FPRAS for MINIMUM ARRIVAL.

Theorem 7. *Let $\varepsilon \in (0, 1)$ and let $\mathcal{G}^{(k)} = (G = (V, E), \{p_e(H_e^{(k)})\})$ be a memory-k stochastic temporal graph such that $p_e(x) \geq \frac{1}{n^c}$ holds for all $x \in \{0, 1\}^k$ and $e \in E$. 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 $1 - \frac{2}{n}$.*

Proof. Let $\mathcal{G}^{(k)}$ be a stochastic temporal graph with two designated vertices s, y. Furthermore let X, as before, be the arrival time of a foremost s-y journey. We will estimate the expectation $\mathbb{E}[X]$ via an unbiased estimator approach. We perform r times independently the following experiment Exp (see Algorithm 3); for now let us assume an arbitrary value for r, to be chosen precisely later.

Algorithm 3 Experiment Exp

Input: A stochastic temporal graph $\mathcal{G}^{(k)}$ on an underlying graph G with n vertices and m edges and two designated vertices s, y of G

1: Starting at time $t = 0$, let $\mathcal{G}^{(k)}$ evolve until time $t' = rn^{c+2}$; the resulting temporal graph has at most $t'm$ time-edges
2: Compute a foremost s-y journey in this temporal graph by running the algorithm of [4] (alternatively, one could run the algorithm of [11])
3: **return** the arrival time of the computed foremost journey

The probability that Exp fails to connect s to y via a journey is equal to the probability that s is not connected to y until time t'. Therefore, Lemma 4 implies that the time to connect s to y exceeds the expectation $\mathbb{E}[X]$ of X by a multiplicative factor of at least rn. By Markov's inequality, this probability of failure is at most $\frac{1}{rn}$. For now, we proceed the analysis of the algorithm assuming that all experiments succeed, and we will take the probability of failure of some experiment(s) into account later on.

Let X_r denote the mean of the outcomes of r executions of the experiment Exp. We note that $\mathbb{E}[X_r] = \mathbb{E}[X]$ meaning that X_r is an unbiased estimator of $\mathbb{E}[X]$. Furthermore, $\sigma(X_r) = \frac{\sigma(X)}{\sqrt{r}}$, and hence by Chebyshev's inequality, for every $\varepsilon \in (0, 1)$ we have:

$$\Pr\left[|X_r - \mathbb{E}[X_r]| \geq \varepsilon \mathbb{E}[X_r]\right] \leq \left(\frac{\sigma(X_r)}{\varepsilon \mathbb{E}[X_r]}\right)^2 \leq \frac{\sigma^2(X)}{r\varepsilon^2 \mathbb{E}^2[X]} \leq \frac{\sigma^2(X)}{r\varepsilon^2}, \tag{12}$$

where the latter inequality follows from the fact that $E[X] \geq 1$. Now, by Lemma 4,

$$\Pr\left[|X_r - \mathbb{E}[X]| \geq \varepsilon \mathbb{E}[X]\right] \leq 2\frac{n^{2c+2}}{\varepsilon^2 r}.$$

Therefore for $r = 2\frac{n^{2c+3}}{\varepsilon^2}$ we have

$$\Pr\left[X_r \in (1-\varepsilon, 1+\varepsilon) \cdot \mathbb{E}[X]\right] \geq 1 - \frac{1}{n},$$

that is, the mean X_r of the outcomes of r executions of the experiment Exp is within a factor of $(1 \pm \varepsilon)$ from $\mathbb{E}[X]$ with probability at least $1 - \frac{1}{n}$.

To analyze the probability of success of our FPRAS, let us call the probability that X_r is *far* from $\mathbb{E}[X]$ "probability of failure of the estimator". Recall that there is also a chance of failure in our algorithm if any of the r experiments fails. Therefore, the probability of success is:

$$\begin{aligned} 1 - \Pr[\text{failure of FPRAS}] &\geq 1 - (\Pr[\text{failure of some Exp}] + \Pr[\text{failure of the estimator } X_r]) \\ &\geq 1 - \left(r\frac{1}{rn} + \frac{1}{n}\right) = 1 - \frac{2}{n}. \end{aligned}$$

Finally, we execute the experiment Exp for $r = 2\frac{n^{2c+3}}{\varepsilon^2}$ times and each execution runs in total for $t' = 2\frac{n^{3c+5}}{\varepsilon^2}$ time. A temporal graph generated in every experiment has at most $t'm$ time-edges, and therefore the algorithm of [4] runs in $O(t'm \log(t'm)) = O(m\frac{n^{3c+5}}{\varepsilon^2} \cdot \log(\frac{n}{\varepsilon}))$ time. Thus the total running time is $O(m\frac{n^{5c+8}}{\varepsilon^4} \cdot \log(\frac{n}{\varepsilon}))$. □

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). To simplify notation, throughout Section 4.1 we write $h(v) \stackrel{\text{def}}{=} h(v,y)$, because the target vertex y is fixed.

We now argue that a best policy for BEST POLICY can be defined, provided the values $h(v)$, for all $v \in V$, are given. To this end, assume that for all $v \in V$, the value $h(v)$ is known; let $v_1 = y, v_2, \ldots, v_n$ be an ordering of vertices of V in non-decreasing values of h (ties broken arbitrarily), namely $h(v_1) \leq h(v_2) \leq \cdots \leq h(v_n)$. Clearly, $v_1 = y$ and $h(v_1) = h(y) = 0$.

Let a_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 $t \in \mathbb{N}$. Intuitively, the best strategy of Alice at time $t+1$ is to look at all neighboring vertices of a_t in G_{t+1} and to find one with minimum h-value, namely a vertex $u \in \arg\min\{h(v) : v \in \Gamma_{G_{t+1}}(a_t)\}$. If $h(u) \geq h(a_t)$, then Alice has no incentive to change vertex and thus $a_{t+1} = a_t$. Otherwise, if $h(u) < h(a_t)$, then $a_{t+1} = u$. We formalize the above in Lemma 5 below. We note that the proof of optimality is in fact a consequence of Lemma 6, but we present here a short proof for the sake of completeness.

Lemma 5. *Assume the values $h(v)$ are given for all $v \in V$, and let $v_1 = y, v_2, \ldots, v_n$ be an ordering of the vertices of V in a non-decreasing ordering with respect to h (ties broken arbitrarily). Then the following policy π is optimal: for any $a_0 \in V, G_1 \subseteq G$,*

$$\pi(a_0, G_1) \stackrel{def}{=} v_{\min\{j : v_j \in \Gamma_{G_1}[a_0]\}}.$$

Proof. For an arbitrary policy π', let $h^{\pi'}(v)$ denote the expected arrival time of an v-y journey that Alice can follow using the policy π'. For the sake of contradiction, suppose that there exists a policy π^* such that, for some vertex $a_0 \in V$ and some snapshot $G_1 \subseteq G$, π^* achieves a better expected arrival time than π, when starting from a_0 in G_1. Without loss of generality we can assume that $a_1 \stackrel{\text{def}}{=} \pi(a_0, G_1) \neq \pi^*(a_0, G_1) \stackrel{\text{def}}{=} a_1^*$ and also that $h^{\pi}(a_1) > h^{\pi^*}(a_1^*)$. But this is a contradiction to the definition of the policy π, since $a_1^* \in \Gamma_{G_1}[a_0]$, and $h(a_1) \leq h(v)$ for all $v \in \Gamma_{G_1}[a_0]$. □

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:

$$h(v_i) = \frac{\sum_{j=1}^{i-1} Q_{i,j} h(v_j) + 1}{1 - Q_i}, \tag{13}$$

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 words, 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 the following bottom-up dynamic programming algorithm[4]:

[4]To avoid trivialities, we assume that the graph induced by the set of edges with non-zero probabilities is connected; in particular, the set of probabilities can be used to model any connected underlying graph G. We also assume that the elements of the list L can be accessed using their index, i.e. L_i is the i-th element of the list.

Algorithm 4 BEST POLICY in memoryless stochastic temporal graphs

Input: A stochastic temporal graph $\mathcal{G}^{(0)} = (G = (V, E), \{p_e\})$.
Output: The values $\{h(v) : v \in V\}$, stored in the ordered list L.

1: Let L be the empty list
2: Append y to L; $h(y) \leftarrow 0$
3: **for** $i = 2$ to n **do**
4: $u \leftarrow \arg\min \left\{ \frac{\sum_{j=1}^{i-1} p_{\{v,L_j\}} \prod_{\ell=1}^{j-1}(1-p_{\{v,L_\ell\}}) h(L_j)+1}{1-\prod_{\ell=1}^{i-1}(1-p_{\{v,L_\ell\}})} : v \notin L, \prod_{\ell=1}^{i-1}(1-p_{\{v,L_\ell\}}) < 1 \right\}$
5: $h(u) \leftarrow \frac{\sum_{j=1}^{i-1} p_{\{u,L_j\}} \prod_{\ell=1}^{j-1}(1-p_{\{u,L_\ell\}}) h(L_j)+1}{1-\prod_{\ell=1}^{i-1}(1-p_{\{u,L_\ell\}})}$
6: Append u to L
7: **return** L and $h(v), v \in V$

Algorithm 4 can be efficiently implemented to run in $O(n^2)$ time and space. This can be achieved by carefully storing intermediate sums and products in step 4; indeed, at step $i+1$ the only new term in the numerator is $p_{\{v,L_i\}} \prod_{\ell=1}^{i-1}(1-p_{\{v,L_\ell\}})h(L_i)$, and in the product $\prod_{\ell=1}^{i}(1-p_{\{v,L_\ell\}})$ which appears in the denominator the only new factor is $1-p_{\{v,L_i\}}$. Concluding, we have the following theorem:

Theorem 8. 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 9. *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 [37]. Recall that 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. Similarly to our reduction in the proof of Theorem 1, 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 2.

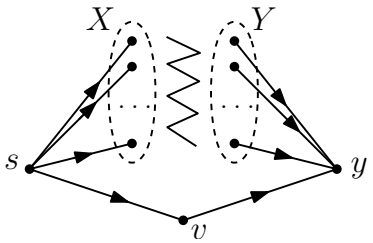

Figure 2: The construction of the DAG H.

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:

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

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

$$2^{n+m}-\frac{\ell}{5} \leq \psi \leq 0.95+2^{n+m}-\frac{\ell}{5}. \tag{14}$$

Thus, 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 $k \geq 1$. We first give a Markov Decision Process (MDP) formulation of our problem under the memory-k model that will be useful for the presentation of our results within the general MDP framework.

4.3.1 An MDP formulation

We follow the notation from chapter 5.4 of [33] to give an MDP formulation. Let $k \geq 1$ be a fixed integer corresponding to the memory of the model. We denote by $\mathcal{I} = V \times (2^G)^k$ the set of *states*, where 2^G denotes the set of spanning subgraphs of the underlying graph G. In particular, each state $(v, H^{(k)}) \in \mathcal{I}$ consists of a vertex v which corresponds to the vertex Alice is on and a sequence of k graphs $H^{(k)}$ corresponding to the k most recent snapshots. For any $t \geq 0$, we will say that $H_t^{(k)} \stackrel{\text{def}}{=} (G_{t-k+1}, G_{t-k+2}, \ldots, G_{t-1}, G_t)$ occurred at time t, if the snapshots at times $t-k+1, t-k+2, \ldots, t-1, t$ are $G_{t-k+1}, G_{t-k+2}, \ldots, G_{t-1}, G_t$, respectively. The set of *actions* for Alice is the set $\mathcal{A} = V$. A *stationary policy* for Alice is a function $f : \mathcal{I} \to \mathcal{A}$ and determines a probability law $\Pr^f$ for a Markov chain $(X_t)_{t \geq 0}$ with values in $\mathcal{I}$ as follows:

(i) Assuming that at time 0 Alice starts from vertex s and the initial sequence of k snapshots is $H_0^{(k)} = (G_{-k+1}, G_{-k+2}, \ldots, G_{-1}, G_0)$, the initial distribution of the Markov chain is given by $\Pr^f \left[X_0 = (s, H_0^{(k)})\right] = 1$, and $\Pr^f \left[X_0 = (v, H^{(k)})\right] = 0$ if $v \neq s$ or $H^{(k)} \neq H_0^{(k)}$.

(ii) For any $t \geq 0$, $v_t \in V$, $v_{t+1} \in V$,

$$\Pr^f \left(X_{t+1} = (v_{t+1}, H_{t+1}^{(k)}) | X_t = (v_t, H_t^{(k)})\right) = \begin{cases} \Pr[G_{t+1} \text{ occurs at } t+1 | H_t^{(k)} \text{ occurred at } t] & \text{if } f(v_t, H_{t+1}^{(k)}) = v_{t+1} \\ 0 & \text{if } f(v_t, H_{t+1}^{(k)}) \neq v_{t+1} \end{cases} \tag{15}$$

Without loss of generality, we will assume that every policy f is *legitimate* in the sense that the following conditions hold:

A. $f(v_t, H_{t+1}^{(k)}) = v_{t+1}$ only if $(v_t, v_{t+1}) \in E(G_{t+1})$, i.e. Alice may visit v_{t+1} in the next step only if G_{t+1} has an edge that connects v_t (the vertex she is currently on) and v_{t+1} (the vertex she wants to go to).

B. Recalling that the goal of Alice is to reach y, we assume that $f(y, H^{(k)}) = y$, for any $H^{(k)}$, i.e. Alice will never leave her target vertex once she reaches it.

We will denote by a_t Alice's t-th action (vertex choice). In particular, $a_0 = s$ and inductively $a_{t+1} = f(a_t, H_{t+1}^{(k)})$, for any $t \geq 0$. Furthermore, let $\mu(G_{t+1} | H_t^{(k)}) \stackrel{\text{def}}{=} \Pr[G_{t+1} \text{ occurs at } t+1 | H_t^{(k)} \text{ occurred at } t]$.

To complete the definition of the Markov Decision Process, we assume that constant cost $c((v, H^{(k)}), a) = 1$ is incurred when action a is chosen in state $(v, H^{(k)})$ with $v \neq y$, otherwise $c((y, H^{(k)}), a) = 0$. Therefore, for a given target y, to every legitimate policy f we can associate an expected total cost $h^f(a_0, y, H_0^{(k)})$ starting from state $(a_0, H_0^{(k)})$, that satisfies: $h^f(y, y, H^{(k)}) = 0$, for

any $H^{(k)}$ and, for any $a_0 \neq y$ and any $H_0^{(k)}$,

$$h^f(a_0, y, H_0^{(k)}) = \mathbb{E}^f\left[\sum_{t=0}^{\infty} c((a_t, H_t^{(k)}), a_{t+1})\right] = \mathbb{E}^f\left[\sum_{t=0}^{\infty} c((a_t, H_t^{(k)}), f(a_t, H_{t+1}^{(k)}))\right] \quad (16)$$

$$= 1 + \mathbb{E}^f\left[\sum_{G_1} \mu(G_1|H_0^{(k)}) \sum_{t=1}^{\infty} c((a_t, H_t^{(k)}), a_{t+1})\right] \quad (17)$$

$$= 1 + \sum_{G_1} \mu(G_1|H_0^{(k)}) h^f(a_1, y, H_1^{(k)}). \quad (18)$$

To be more clear, the expectations in equation (16) are over random variables $G_1, G_2, \ldots$, while the expectation in equation (17) is over $G_2, G_3, \ldots$. Furthermore, equation (17) follows by conditioning on G_1 and equation (18) follows by observing that, by symmetry, $\mathbb{E}^f\left[\sum_{t=1}^{\infty} c((a_t, H_t^{(k)}), a_{t+1})\right]$ equals the expected total cost starting from $(a_1, H_1^{(k)})$.

Note also that the values of the variables a_1 and $H_1^{(k)}$ depend on G_1, so they can be different in different terms of the sum over all possible G_1 in equation (18); indeed, $H_1^{(k)}$ is obtained from $H_0^{(k)}$ by appending G_1, and a_1 is equal to $f(a_0, H_1^{(k)})$.

Observation 2. Any *policy f guiding Alice from s to y must satisfy recurrence (18), with initial condition $h^f(y, y, H^{(k)}) = 0$, for every $H^{(k)}$.*

Our objective is to find a policy that minimizes the expected total cost $h^f(a_0, y, H_0^{(k)})$. In particular, this policy will have the value $h^*(a_0, y, H_0^{(k)}) = \inf_f h^f(a_0, y, H_0^{(k)})$ which will be equal to the expected arrival time of a journey suggested to Alice by an optimal policy. In fact, without loss of generality we will assume that the h^*-values of an optimal policy satisfy $h^*(a_0, y, H_0^{(k)}) = \inf_f h^f(a_0, y, H_0^{(k)})$, for all $a_0 \in V$ and all $H_0^{(k)} = (G_{-k+1}, G_{-k+2}, \ldots, G_{-1}, G_0)$, such that $G_i \subseteq G$, for all $-k+1 \leq i \leq 0$.

4.3.2 A doubly exponential-time algorithm

We now provide our doubly exponential-time algorithm for Best Policy in the memory-k model, where $k \geq 1$. In order to simplify the notation and presentation of this section, we only provide the proof of the algorithm for the special case $k = 1$; the analysis for arbitrary $k \geq 1$ carries then easily over, as we discuss at the end of the section.

Memory-1 case. Following the notation of Section 4.3.1 for memory-1, we denote by $\mu(G''|G')$ the probability that the next snapshot is G'', given that the current snapshot is G'. Furthermore, for $a_0 \in V$, let $h(a_0, y, G_0)$ be the expected arrival time of a journey from a_0 to y suggested to Alice by an optimal policy, given that the starting graph instance is equal to G_0.

Similarly to the memory-0 case, we first show that we can define an optimal policy provided all h-values are known in advance. In particular, we define the following policy π: For any time step $t \geq 0$, if at time t Alice was on a vertex a_t and at time $t+1$ the graph instance is G_{t+1}, then at time $t+1$ she will move to a vertex $u \in \Gamma_{G_{t+1}}[a_t]$ that has minimum $h(u, y, G_{t+1})$, that is,

$$\pi(a_t, G_{t+1}) \stackrel{\text{def}}{=} a_{t+1} \in \arg\min\left\{h(u, y, G_{t+1}) : u \in \Gamma_{G_{t+1}}[a_t]\right\}. \quad (19)$$

Lemma 6. *Policy π is optimal.*

Proof. The lemma follows by the definition of the policy π in (19) and by part (ii) of Theorem 5.4.3 of [33]. □

Notice that in the definition of π, we assumed that the h-values are given. Therefore, to determine π we need to compute $h(a_0, y, G_0)$, for every $a_0 \in V$ and $G_0 \subseteq G$. We start by rewriting recurrence (18) for policy π:

$$h^\pi(a_0, y, G_0) = 1 + \sum_{G_1} \mu(G_1|G_0) h^\pi(\pi(a_0, G_1), y, G_1). \quad (20)$$

Since π is optimal, the left hand side of the above equation is equal to $h(a_0, y, G_0)$. Furthermore, by definition of $\pi(a_0, G_1)$,

$$h^{\pi}(\pi(a_0, G_1), y, G_1) = \min\{h(u, y, G_1) : u \in \Gamma_{G_1}[a_0]\}.$$

Therefore, recurrence (20) becomes

$$h(a_0, y, G_0) = 1 + \sum_{G_1} \mu(G_1|G_0) \min\{h(u, y, G_1) : u \in \Gamma_{G_1}[a_0]\}. \tag{21}$$

The difference of the above recurrence with the one we derived for the memory-0 case in (13) is that here we need to get rid of the minimum in the right hand side. To this end, suppose that we know an *ordering* of the triplets $(a_0, y, G_0), a_0 \in V, G_0 \subseteq G$, non-decreasingly according to the values $h(a_0, y, G_0)$, breaking ties arbitrarily. Notice that these are $n2^m \stackrel{\text{def}}{=} N$ such values, where $m = |E|$ is the number of edges of G. Then the minimum in recurrence (21) can be replaced with the corresponding h-value, which is completely determined by the graph G_1 and the vertex a_0. Doing this for all different vertices a_0 and graphs G_0, we get a linear system with N equations coming from (21) and N variables (the h-values). To this system, we then add the initial conditions $h(y, y, G_0) = 0$, for all $G_0 \subseteq G$. One solution of this system (if it exists) can be found in $O(N^3)$ time.

Therefore, one approach for computing all the h-values (which however is not necessarily guaranteed to always work, as we describe below), is the following naive brute-force algorithm: For each of the (at most) $N!$ orderings of the triplets $(u_0, y, G_0), a_0 \in V, G_0 \subseteq G$, solve the linear system derived by the recurrence (21) as described above, assuming the ordering is "correct", namely it corresponds to an ordering in increasing values of $h(a_0, y, G_0)$. Then check if the ordering we get from the solution to that system is the same as the one we assumed. If not, then consider a different ordering.

Notice however the following correctness issue with the above naive brute-force algorithm: suppose the correct ordering σ^* is considered and we construct the corresponding linear system of equations (call it Σ) based on (21). Clearly, Σ has at least one solution, but what happens if there are more than one solutions, some of which giving an ordering that is not consistent with σ^*? Can we find the correct solution among all other solutions of Σ? To circumvent this problem, we replace the linear system of equalities with a linear system of inequalities (constraints). To this end, let $u^{\sigma^*}_{a_0,G_1}$ be the vertex such that $u^{\sigma^*}_{a_0,G_1} \in \Gamma_{G_1}[a_0]$ and the triplet $(u^{\sigma^*}_{a_0,G_1}, y, G_1)$ appears in σ^* before all triplets (u, y, G_1), for all $u \in \Gamma_{G_1}[a_0] \setminus \{u^{\sigma^*}_{a_0,G_1}\}$. We want to find *any* solution satisfying the following constraints:

$$h'(a_0, y, G_0) = 1 + \sum_{G_1} \mu(G_1|G_0) h'(u^{\sigma^*}_{a_0,G_1}, y, G_1) \tag{22}$$

$$h'(u^{\sigma^*}_{a_0,G_1}, y, G_1) \leq h'(u, y, G_1), \quad \forall u \in \Gamma_{G_1}[a_0], a_0 \in V \setminus \{y\}, G_0 \subseteq G \tag{23}$$

$$h'(a_0, y, G_0) \geq 0 \quad \forall a_0 \in V, G_0 \subseteq G \tag{24}$$

$$h'(y, y, G_0) = 0 \quad \forall G_0 \subseteq G. \tag{25}$$

By definition of σ^*, the above set of constraints has at least one solution, namely the one corresponding to the h-values of an optimal policy. In Theorem 11 we prove that this is the only feasible solution. For the proof, we also need the following Theorem from [33], which we restate here in our notation for convenience:

Theorem 10 (Policy increment, Theorem 5.4.4, [33])**.** *Given one stationary policy f, let θf denote the policy that, for every a_0, G_0 minimizes $\sum_{G_1} \mu(G_1|G_0) h^f((\theta f)(a_0, G_1), y, G_1)$. Then, for all a_0, G_0*

$$\lim_{k \to \infty} h^{\theta^k f}(a_0, y, G_0) = h(a_0, y, G_0), \tag{26}$$

provided $\mathbb{E}_{(a_0,G_0)} h^f(a_n, y, G_n) \to 0$ as $n \to \infty$.

In the above, the notation $\theta^k f$ means the application of policy increment k times. Furthermore, in the expectation $\mathbb{E}_{(a_0,G_0)} h^f(a_n, y, G_n)$, the state (a_n, G_n) is a random variable and its distribution is determined by an optimal policy, given that we start at (a_0, G_0). We note that, the condition of the Theorem holds in our case by transience of the underlying Markov chain (i.e. once Alice reaches y she does not leave and no further cost is incurred after that).

We now prove the following:

Theorem 11. *Let σ^* be an ordering of the triplets $(a_0, y, G_0), a_0 \in V, G_0 \subseteq G$, in increasing order of h-values. Let $(h'^*(a_0, y, G_0) : a_0 \in V, G_0 \subseteq G)$ be any feasible solution to the set of constraints (22) to (25). Then $h'^*(a_0, y, G_0) = h(a_0, y, G_0)$, for all a_0, G_0.*

Proof. We define the following policy π^* (similar to the definition of π earlier, but using the h'^*-values instead of the h-values): For any time step $t \geq 0$, if at time t Alice was on a vertex a_t and at time $t+1$ the graph instance is G_{t+1}, then at time $t+1$ she will move to a vertex $u \in \Gamma_{G_{t+1}}[a_t]$ that has minimum $h'^*(u, y, G_{t+1})$, that is,

$$\pi^*(a_t, G_{t+1}) \stackrel{\text{def}}{=} a_{t+1} \in \arg\min \left\{ h'^*(u, y, G_{t+1}) : u \in \Gamma_{G_{t+1}}[a_t] \right\}. \tag{27}$$

Since the h'^*-values satisfy constraint (22) and also, by constraint (23), $u^{\sigma^*}_{a_0,G_1} \in \arg\min\{h'^*(u, y, G_1) : u \in \Gamma_{G_1}[a_0]\}$, the expected arrival time of a journey from a_0 to y when Alice follows policy π^* is equal to $h'^*(a_0, y, G_0)$, for all $a_0 \in V$ and $G_0 \subseteq G$.

Observe that, if π^* is not optimal, then we can successively apply policy increments $\theta^k\pi^*, k \to \infty$ as in Theorem 10, to eventually reach optimality. Notice also that the sum in the definition of policy increment for π^* can be written as

$$\sum_{G_1} \mu(G_1|G_0) h^{\pi^*}((\theta\pi^*)(a_0, G_1), y, G_1) = \sum_{G_1} \mu(G_1|G_0) h'^*((\theta\pi^*)(a_0, G_1), y, G_1). \tag{28}$$

Therefore, by definition, π^* itself is a policy that minimizes the above sum, and so we can take $\theta\pi^* = \pi^*$. Consequently, no improvement by increment is possible, implying that π^* is optimal. In particular, $(h'^*(a_0, y, G_0) : a_0 \in V, G_0 \subseteq G)$ is the same as $(h(a_0, y, G_0) : a_0 \in V, G_0 \subseteq G)$, and the proof is completed. □

In view of the above, our algorithm for the memory-1 case is Algorithm 5.

Algorithm 5 BEST POLICY in memory-1 stochastic temporal graphs

Input: A stochastic temporal graph $\mathcal{G}^{(1)}$.
Output: The values $(h(a_0, y, G_0) : a_0 \in V, G_0 \subseteq G)$.

1: **for** all orderings σ^* of the triplets $(a_0, y, G_0), a_0 \in V, G_0 \subseteq G$ **do**
2: find a feasible solution $(h'(a_0, y, G_0) : a_0 \in V, G_0 \subseteq G)$ to the linear program (22)-(25)
3: **if** $(h'(a_0, y, G_0) : a_0 \in V, G_0 \subseteq G)$ is consistent with σ^* **then**
4: **return** $(h'(a_0, y, G_0) : a_0 \in V, G_0 \subseteq G)$

The set of constraints (22) to (25) has $N = n2^m$ variables, namely $\{h'(a_0, y, G_0) : a_0 \in V, G_0 \subseteq G\}$. Furthermore, there are $(n-1)2^m$ constraints of the form (22), at most $n^2 2^m$ constraints of the form (23) and $n2^m$ non-negativity and initialization constraints, i.e. $O(nN)$ constraints in total. Therefore, Vaydia's algorithm for linear programming [40] can find an optimum solution in $O((nN)^{2.5})$ time. Since we need to solve this set of constraints for every possible ordering of the N different triplets (a_0, y, G_0), our brute-force approach runs in $O(N!\ (nN)^{2.5}) = O(N^N)$ time.

Memory-k case ($k \geq 1$). The above analysis for the memory-1 model directly carries over to the memory-k model, for any $k \geq 1$. Indeed, the correctness proof can be slightly modified by replacing everywhere the subgraphs G_t of G by the length-k histories $H_t^{(k)}$, respectively. Furthermore, the running time analysis carries over to the case of an arbitrary $k \geq 1$ by replacing $N = n2^m$ by $N' = n2^{km}$. Summarizing, we obtain the following theorem.

Theorem 12. *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.*

Remark 1. *It is easy to see that the running time of the above brute-force algorithm is dominated by the number of different orderings $N'!$, and thus we have a doubly exponential algorithm (recall that $N' = n2^{km}$). A different approach that can potentially lead to a faster algorithm is to start from an arbitrary initial policy and successively apply policy increments as in Theorem 10. Even though the*

convergence analysis of such an approach is non-trivial, one could use it to find the optimal ordering σ^* fast[5] *and then use σ^* to find the unique solution to the set of constraints (22)-(25).*

5 Acknowledgements

This work was funded in part by:

(i) the NeST initiative of the EEE/CS School of the University of Liverpool,

(ii) the EPSRC grants on "Algorithmic Aspects of Temporal Graphs", EP/P020372/1 and EP/P02002X/1,

(iii) Greece and the European Union (European Social Fund - ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning" in the context of the project "Reinforcement of Postdoctoral Researchers" (MIS-5001552), implemented by the State Scholarships Foundation (IKY), and

(iv) the EPSRC-funded AlgoUK network, EP/R005613/1.

References

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

[2] E. Akrida, G. Mertzios, P. 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.

[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] A. Anagnostopoulos, J. Lacki, S. Lattanzi, S. Leonardi, and M. Mahdian. Community detection on evolving graphs. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS)*, pages 3522–3530, 2016.

[6] 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.

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

[8] P. Basu, S. Guha, A. Swami, and D. Towsley. Percolation phenomena in networks under random dynamics. In *Proceedings of the 4th International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–10, 2012.

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

[10] P. Basu, F. Yu, M. P. Johnson, and A. Bar-Noy. Low expected latency routing in dynamic networks. In *Proceedings of the 11th IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 267–271, 2014.

[5] This is possible if after a relatively small (say polynomial in N') number of steps the ordering does not change.

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

[12] A. Casteigts and P. Flocchini. Deterministic Algorithms in Dynamic Networks: Formal Models and Metrics. Technical report, Defence R&D Canada, April 2013.

[13] A. Casteigts and P. Flocchini. Deterministic Algorithms in Dynamic Networks: Problems, Analysis, and Algorithmic Tools. Technical report, Defence R&D Canada, April 2013.

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

[16] 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.

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

[18] J. Enright, K. Meeks, G. Mertzios, and V. Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. In *Proceedings of the 44th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 57:1–57:15, 2019.

[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*, 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 *Proceedings of the Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 201–210, 2018.

[24] A. 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 *Proceedings of the 32nd Annual ACM symposium on Theory of computing (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. Mertzios, O. Michail, I. Chatzigiannakis, and P. Spirakis. Temporal network optimization subject to connectivity constraints. *Algorithmica*, pages 1416–1449, 2019.

[30] G. Mertzios, H. Molter, and V. Zamaraev. Sliding window temporal graph coloring. In *Proceedings of the 33st AAAI Conference on Artificial Intelligence (AAAI)*, pages 7667–7674, 2019.

[31] O. Michail and P. Spirakis. Elements of the theory of dynamic networks. *Communications of the ACM*, 61(2):72–72, Jan. 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] J. Norris. *Markov chains*. Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, 1998.

[34] R. Ogier and V. Rutenburg. Minimum-expected-delay alternate routing. In *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, pages 617–625, 1992.

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

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

[37] J. Provan and M. 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.

[38] N. Santoro. Computing in time-varying networks. In *Proceedings of the 13th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, page 4, 2011.

[39] C. Scheideler. Models and techniques for communication in dynamic networks. In *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 27–49, 2002.

[40] P. Vaidya. Speeding-up linear programming using fast matrix multiplication. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 332–337, 1989.

[41] J. Valdes, R. E. Tarjan, and E. L. Lawler. The recognition of series parallel digraphs. *SIAM Journal on Computing*, 11(2):298–313, 1982.

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