# Dynamic Network Discovery via Infection Tracing

**Ben Bals**[1,2], **Michelle Döring**[3], **Nicolas Klodt**[3] and **George Skretas**[3]
[1]Centrum Wiskunde & Informatica, Amsterdam
[2]Vrije Universiteit Amsterdam
[3]Hasso Plattner Institute, Potsdam
bjjb@cwi.nl, {michelle.doering, nicolas.klodt, georgios.skretas}@hpi.de,

## Abstract

Researchers, policy makers, and engineers need to make sense of data from spreading processes as diverse as rumor spreading in social networks, viral infections, and water contamination. Classical questions include predicting infection behavior in a given network or deducing the network structure from infection data. Most of the research on network infections studies *static graphs*, that is, the connections in the network are assumed to not change. More recently, *temporal graphs*, in which connections change over time, have been used to more accurately represent real-world infections, which rarely occur in unchanging networks. We propose a model for temporal graph discovery that is consistent with previous work on static graphs and embraces the greater expressiveness of temporal graphs. For this model, we give algorithms and lower bounds which are often tight. We analyze different variations of the problem, which make our results widely applicable and it also clarifies which aspects of temporal infections make graph discovery easier or harder. We round off our analysis with an experimental evaluation of our algorithm on real-world interaction data from the Stanford Network Analysis Project and on temporal Erdős-Renyi graphs. On Erdős-Renyi graphs, we uncover a threshold behavior, which can be explained by a novel connectivity parameter that we introduce during our theoretical analysis.

## 1 Introduction

Predicting the spread of infections requires precise knowledge about the network in which they take place. These spreading processes can be vastly different; they involve everything from diseases, misinformation, marketing material to contaminants in sewage networks. All of these can be modeled in a similar fashion, and we can thus utilize a common algorithmic toolkit for their analysis. Most famously, the *influence maximization problem*, introduced by [Kempe *et al.*, 2003], wants to find which node in a network should be infected to maximize the number of nodes infected by the spreading process. Other problems include finding a sensor placement to detect outbreaks as quickly as possible [Leskovec *et al.*, 2007], or to estimate the source (of an infection or rumor) from data about the spreading process [Berenbrink *et al.*, 2023; Luo *et al.*, 2013; Zhu and Ying, 2016].

One common assumption is that the underlying network is known. However, this is not true in many real-world scenarios, and thus, we require algorithms for *network discovery*. While discovering networks is a fundamental problem in data mining, which can be approached from different angles [Rong *et al.*, 2016; Park and Honorio, 2016], one natural approach is to discover the underlying network from the infection data itself. This idea has been extensively studied [Pouget-Abadie and Horel, 2015; Lokhov, 2016; Daneshmand *et al.*, 2014; Netrapalli and Sanghavi, 2012]. In particular, [Chistikov *et al.*, 2024] consider a model where the party wishing to discover the network may even intervene in the spreading process, e.g., by publishing a social media post and watching its spread. Beyond this application, network discovery is an interesting and relevant problem in its own right. After having discovered the network from infection data, we are free to abstract away from the spreading process and use the resulting network in a host of different ways. This is especially relevant for the study of social networks, both real-world and online, where infection data can reveal underlying structures that are otherwise difficult to observe.

To the best of our knowledge, every paper that studies network discovery makes the same simplifying assumption: the underlying network is *static*. That is, the connections of the network do not change over time. In most applications, that is not a realistic assumption. For example, if two people are linked in an in-person social network, that does not imply that a disease can spread from one to the other at every point in time, but only when they physically meet. Motivated by this fact, researchers have begun to study the classical infection analysis tasks on *temporal graphs*.

Temporal graphs are a model of dynamic networks where the edges only exist at some time steps. This model has received considerable attention from theoretical computer scientists for both foundational problems [Michail, 2016; Danda *et al.*, 2021; Casteigts *et al.*, 2021a; Akrida *et al.*, 2019] and a growing number of applications, including social networks [Casteigts *et al.*, 2021b]. For our purposes, a temporal graph $\mathcal{G} = (V, E, \lambda)$ with lifetime $T_{\max} \in \mathbb{N}$ is a static graph $G = (V, E)$ with a function $\lambda \colon E \to 2^{[T_{\max}]}$ indicat-

ing that edge $e \in E$ exists precisely at the time steps $\lambda(e)$. [Gayraud *et al.*, 2015], are the first to study the influence maximization problem on temporal graphs under the independent cascade model (introduced by [Kempe *et al.*, 2003]). [Deligkas *et al.*, 2024] build on this work and analyze the influence maximization problem on temporal graphs under the SIR model (a standard biological spreading model closely related to the susceptible-infected-resistant model [Hethcote, 1989]). However, no work on network discovery on temporal graphs has been conducted yet.

**Our Contribution** is twofold: (i) we define the temporal network discovery problem as a round-based, interactive two-player game, (ii) we provide algorithms and lower bounds for different parameters of the network discovery game, and validate our results both with theoretical proofs, as well as experiments conducted on real-world and synthetic networks. *Statements where proofs or details are omitted to the full version due to space constraints are marked with* $\star$.

In Section 3, we define the two-player game. In each round, the *Discoverer* (abbreviated $\mathcal{D}$) initiates infections and observes the resulting infection chains, aiming to identify the time labels of all edges in as few rounds as possible. The *Adversary* (abbreviated $\mathcal{A}$) gets to pick the shape and temporal properties of the graph, with the aim of forcing $\mathcal{D}$ to take as long as possible to accomplish their task.

In Section 4, we provide the `DiscoveryFollow` algorithm, which solves the graph discovery problem in $6|E(\mathcal{G})| + |EC_\delta(\mathcal{G})|\lceil T_{\max}/\delta\rceil$ rounds, where $|EC_\delta(\mathcal{G})|$ are the $\delta$-edge connected components. Intuitively, this is a grouping of the edges such that only edges from the same component may influence each other during infection chains. In Section 5, we prove that the running time of the `DiscoveryFollow` algorithm is asymptotically tight in the number of edges. Formally, we prove there is an infinite family of graphs such that any algorithm winning the graph discovery game requires at least $\Omega(|E(\mathcal{G})|)$ rounds. Crucially, this cannot be improved even if $\mathcal{D}$ is allowed to start multiple infection chains per round. We also prove that there is an infinite family of graphs such that the minimum number of rounds required to win the graph discovery game grows in $\Omega(nT_{\max}/(\delta k))$, where $k$ is the number of infection chains $\mathcal{D}$ may start per round.

We finish our theoretical analysis in Section 6, where we explore variations of the graph discovery problem. We analyze the case where the feedback $\mathcal{D}$ receives about the infection chains is reduced to infection times. Surprisingly, we are able to show that our `DiscoveryFollow` algorithm directly translates to this scenario. We also discuss what happens if $\mathcal{D}$ has no information about the static graph in which the infections are taking place. Third, we allow the temporal graph to now contain multiedges or more than one label per edge.

In Section 7, we empirically validate our theoretical results. Using both synthetic and real-world data, we execute the `DiscoveryFollow` algorithm and observe its performance. We utilize the natural temporal extension of Erdős-Renyi graphs [Casteigts *et al.*, 2022] as well as the `comm-f2f-Resistance` data set from the Stanford Large Network Dataset Collection [Kumar *et al.*, 2021], a social network of face-to-face interactions. Beyond the running time, we closely analyze which factors affect the performance of the algorithm. We see that the density of the graph affects the performance since, in dense graphs, it needs to spend less time finding new $\delta$-edge connected components. On Erdős-Renyi graphs, we provide evidence that this effect is mediated by the number of $\delta$-edge connected components, which exhibits a threshold behavior in $T_{\max}/(|V(\mathcal{G})|\, p)$, where $p$ is the Erdős-Renyi density parameter. This prompts us to give a conjecture on this threshold behavior, which mirrors the famous threshold behavior in the connected components of nodes in static Erdős-Renyi graphs [Erdos *et al.*, 1960].

## 2 Preliminaries

For $n \in \mathbb{N}^+$, $x \in \mathbb{N}$, let $[x, n] \coloneqq \{x, \dots, n\}$ and $[n] \coloneqq [1, n]$.

A *temporal graph* $\mathcal{G} = (V, E, \lambda)$ with lifetime $T_{\max}$ is composed of an undirected *(underlying) static graph* $G = (V, E)$ together with a *labeling function* $\lambda\colon E \to 2^{[T_{\max}]}$, denoting $e \in E$ being present precisely at time steps $\lambda(e)$. We also write $V(\mathcal{G})$ for the nodes of $\mathcal{G}$, and $E(\mathcal{G})$ for its edges. A *temporal path* a sequence of timed edges $(e_1, t_1), \dots, (e_\ell, t_\ell)$ that forms a path in $G$ with strictly increasing labels, i. e., for all $i \in [\ell]$, $t_i < t_{i+1}$ and $t_i \in \lambda(e_i)$. Apart from Section 6, we consider *simple* temporal graphs where each edge has exactly one label. Abusing notation, we therefore also use $\lambda$ as if it were defined as $\lambda\colon E \to [T_{\max}]$, and regard temporal paths as the corresponding sequence of nodes.

We use the susceptible-infected-resistant (SIR) model, in which a node is either in a *susceptible*, *infected*, or *resistant* state. This model of temporal infection behavior is based on [Deligkas *et al.*, 2024] (and more historically flows from [Kermack *et al.*, 1927] and [Pastor-Satorras *et al.*, 2015]). An *infection chain* in the SIR model unfolds as follows. At most $k \in \mathbb{N}$ nodes may be infected by $\mathcal{D}$ at arbitrary points in time, which we call *seed infections* denoted as $S \subseteq V \times [0, T_{\max}]$. Otherwise, a node $u$ becomes *infected at time step* $t$ if and only if it is susceptible and there is a node $v$ infectious at time step $t$ with an edge $uv$ with label $t$. Then $u$ is infectious from time $t + 1$ until $t + \delta$, after which $u$ becomes resistant. Note that if a susceptible node has two or more infected neighbors at the same time, it can be infected by any one of them, but only one. Thus, a given set of seed infections may result in multiple possible infection chains.

The *infection log* of an infection chain records which node was infected by which neighbor at what time. Formally, the infection log is a set $L \subseteq V^2 \times [0, T_{\max}]$, where $(u, v, t) \in L$ indicates that $u$ infected $v$ at time step $t$. A seed infection at $u$ at time $t$ is denoted by $(u, u, t) \in L$. The *infection timetable* $T \subseteq V \times [0, T_{\max}]$ records only when a node became infected, omitting which neighbor caused the infection. We call an infection log $L$ *consistent* with a given set of seed infections $S$ if there is an infection chain seeded with $S$ that produces $L$. Consistency for infection timetables is defined analogously.

While multiple consistent infection logs may exist for a set of seeds, there is exactly one consistent infection timetable.

$\star$ **Lemma 1.** *Let $S \subseteq V \times [0, T_{\max}]$ be a set of seed infections, and $L_1$, $L_2$ be two infection logs consistent with $S$. Then the induced infection timetables $T_i \coloneqq \{(v, t) \mid (u, v, t) \in L_i\}$ (for $i = 1, 2$) are the same, that is, $T_1 = T_2$.*

## 3 Modeling Temporal Graph Discovery

We model temporal graph discovery (TGD) as a game, where the Discoverer $\mathcal{D}$ seeks to uncover information about the graph (e.g., edges and their time labels), while the Adversary $\mathcal{A}$ influences the infection behavior (e.g., by assigning edge labels) with the goal of delaying $\mathcal{D}$'s progress. See Figure 1 for a description of the structure of the game.

---

**Input:** $T_{\max}, \delta, k, n \in \mathbb{N}$

1. $\mathcal{D}$ learns the nodes and possibly additional information.
2. In each round, $\mathcal{D}$ submits up to $k$ seed infections $S$ and $\mathcal{A}$ responds with an infection log consistent with $S$.
3. To end the game, $\mathcal{D}$ submits a temporal graph $\mathcal{G}$ to $\mathcal{A}$. $\mathcal{A}$ responds with a temporal graph $\mathcal{G}'$ consistent with all infection logs. If $\mathcal{G}' \neq \mathcal{G}$, $\mathcal{A}$ wins. Otherwise, $\mathcal{D}$ wins.

---

Figure 1: The temporal graph discovery game

As default, we assume $\mathcal{D}$ learns the static edges in Step 1. This is a strong, but convenient assumption, and we later show that many of our results translate to other variations.

We measure the quality of a $\mathcal{D}$ strategy by the number of rounds it needs to win the game.

**Definition 1.** *For parameters $T_{\max}$, $\delta$, $k$ and a static graph $G$, define the* graph discovery complexity *as the minimum number of rounds required for $\mathcal{D}$ to win any TGD game.*

We start off with the simplest algorithm: brute-forcing all combinations of nodes $v \in V(\mathcal{G})$ and time steps $t \in [T_{\max}]$ as seed infections. This will serve as a baseline for comparing more sophisticated algorithms and possible lower bounds.

⋆ **Theorem 1.** *There is an algorithm that wins the TGD game in $|V(G)|\, T_{\max}$ rounds.*

## 4 An Algorithm for Graph Discovery

Before proposing a better algorithm for TGD, we consider a simpler problem: finding an ideal patient zero.

**Definition 2.** *A node $v$ is an* ideal patient zero (IPZ) with time $t$ *if seed-infecting $\{(v,t)\}$ causes every node to become infected. We call $(v,t)$ an* IPZ pair.

We adapt the TGD game. $\mathcal{D}$ submits either a pair $(u,t)$, representing its guess for an IPZ, or $\perp$ if it believes none exists. $\mathcal{A}$ responds with a graph $\mathcal{G} = (V, E, \lambda)$ consistent with all infection logs. $\mathcal{A}$ wins if $\mathcal{D}$'s guess is incorrect or if $\perp$ is submitted when an IPZ exists.

Observe that testing for each node if at every time it could spread an infection $(u, \lambda(u,v) - 1)$ for every neighbor $v$, does not always find an existing patient zero. Instead, brute-forcing every combination as in Theorem 1, discovers every IPZ pair in $T_{\max}\,|V(\mathcal{G})|$ rounds.

**Corollary 1.** *There is a strategy for $\mathcal{D}$ that can win the IPZ game in $T_{\max}\,|V(G)|$ rounds.*

The number of rounds required by the algorithm is trivially bad in its efficiency, as it is brute-forcing the entire graph instead of using the temporal graph's structure at all.

Note, two edges can only participate in the same infection chain if they are connected by a series of edges with time labels differing by at most $\delta$. This leads to a new connectivity parameter for temporal graphs, reflecting the constraint that nodes remain infectious for only a limited time, and thus, infection chains must also respect these timing constraints.

**Definition 3.** *Let $\mathcal{G}$ be a temporal graph and $\delta \in \mathbb{N}^+$. Consider the relation linking two edges if their time difference at a shared endpoint is at most $\delta$. Let $EC_\delta$ be the partitioning of edges obtained by taking the transitive closure of this relation. We call the resulting equivalence classes the* $\delta$-edge connected components *(or $\delta$-ecc's for short) of $\mathcal{G}$.*

Any infection chain caused by a single seed node (e.g., an IPZ) must be contained in a single $\delta$-ecc. This motivates Algorithm 1. See Figure 2 for an illustration of an execution.

---

**Algorithm 1:** `Follow` discovers the neighbors of $v_0$. `Explore` discovers their respective $\delta$-ecc's.

---

**fun** `Follow(`$G$, $\delta$`)`**:**

1. Pick node $v_0 \in V(G)$ arbitrarily.
2. For $i \in [0, \lceil T_{\max}/\delta \rceil]$, perform a round with seed infection $(v_0, i\delta)$.
3. For each edge $e = v_0 u$ that successfully infects: `Explore(`$e, \lambda(e)$`)`.

**fun** `Explore(`$u$, $t$`)`**:**

1. For each $t' \in \{t - \delta - 1, t - 1, t\}$: If there has been no round with seed infection $(u, t')$, seed an infection $(u, t')$. Store that this has been done.
2. For each newly infected edge $uv$ with infection time $t$: `Explore(`$v$,$t$`)`.

---

The next result is the crucial property allowing us to argue that Algorithm 1 does not miss relevant edges.

**Corollary 2.** *If* `Follow` *discovers an edge from a $\delta$-ecc, it discovers the whole component.*

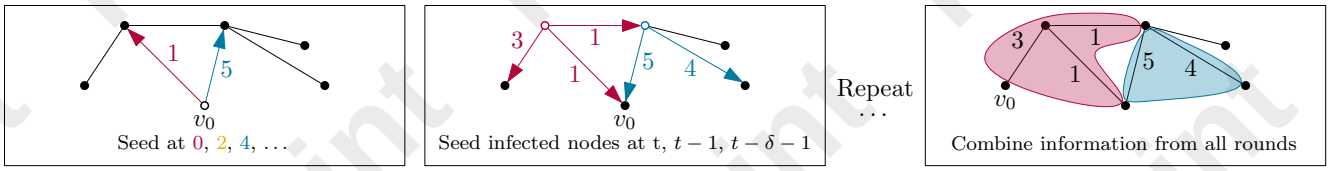
This tool in hand, we prove the correctness of Algorithm 1, which requires at most $6\,|E(G)| + \lceil T_{\max}/\delta \rceil$ rounds.

⋆ **Theorem 2.** *The* `Follow` *algorithm (Algorithm 1) correctly solves the IPZ problem.*

*Proof sketch.* Observe that Step 2 always finds all edges adjacent to $v_0$ and by Corollary 2, the `Explore` algorithm discovers the whole $\delta$-ecc's of these edges. Finally, we show that if an IPZ exists, their infection chain must be subset of one of these components. □

We now extend this idea to obtain a better TGD algorithm, see Algorithm 2. Note that its running time does not only depend on the static, but also the temporal structure of the graph. Recall that `Follow` explores precisely the $\delta$-ecc's adjacent to the start node $v_0$ and note that a graph is discovered if and only if all its $\delta$-ecc's are discovered.

⋆ **Theorem 3.** *Algorithm 2 wins any instance of the TGD game on $\mathcal{G}$ in at most $6\,|E(\mathcal{G})| + |EC_\delta(\mathcal{G})|\,\lceil T_{\max}/\delta \rceil$ rounds.*




1. Finding components at $v_0$ 2. Explore components Discovered edge labels and components

Figure 2: An execution of the `Follow` algorithm. In this example, $\delta = 2$. We perform the initial search for an edge at $v_0$. Ringed nodes indicate seed infections. Next, we explore the two nodes we found in the initial search by performing seed infections at $t$, $t-1$, and $t-\delta-1$ where $t$ is the time step we observed the node being infected. We repeat that until we have no such seed infections to perform without repetitions. Since no $\delta$-ecc spans all nodes, we conclude that no IPZ pair exists.

**Algorithm 2:** The TGD extension of `Follow`.

**fun** `DiscoveryFollow`$(G, \delta)$**:**
  **while** *there is a node* $v_0$ *with an adjacent edge for which the label is still unknown* **do**
    In steps of size $\delta$ perform rounds such that $v_0$ is seed-infected at $0, \delta, 2\delta, \ldots, T_{\max}$.
    For each edge $e = v_0u$ which successfully infects from $v_0$ to a neighbor:
    `Explore`$(u, \lambda(e))$.

## 5 Lower Bounds for Graph Discovery

We build a toolkit for proving lower bounds in the TGD game. Initially, every edge could have any label. As the game progresses, information is revealed to $\mathcal{D}$. A successful infection attempt determines the edge's label, while an unsuccessful attempt where one endpoint remains susceptible reduces the possible labels by at least one. With this, we can use a potential argument to establish lower bounds on TGD complexity.

⋆ **Theorem 4.** *Let $\mathcal{G}$ be a temporal graph and $T_{\max}$, $\delta$, $k$ parameters for the TGD game. For a sequence of seed infection sets $S_1, \ldots, S_a$, define $\Phi(i)$ as the sum of the sizes of the sets of consistent labels over all edges after rounds 1 to $i$. Then, $S_1, \ldots, S_a$ is a* witnessing schedule *if $\Phi(a) = |E(\mathcal{G})|$.*

Recall that the brute-force algorithm for TGD takes $\mathcal{O}(nT_{\max})$ rounds (Theorem 1). In the worst case, this can be improved by at most a factor of $\delta k$.

**Theorem 5.** *For all $\delta, k \in \mathbb{N}^+$ and $T_{\max} \geq 4$, there is an infinite family of temporal graphs $\{\mathcal{G}_n\}_{n\in\mathbb{N}}$ such that each $\mathcal{G}_n$ has $\Theta(n)$ nodes and the minimum number of rounds required to win the TGD game on it grows in $\Omega(n(T_{\max}-3)/(\delta k))$.*

*Proof.* Let $n, \delta \in \mathbb{N}^+$ and $T_{\max} \geq 4$, with $n$ even for simplicity. We construct a temporal graph $\mathcal{G}_n$ on vertices $v_1, \ldots, v_n$. The edges are as follows (with some edge labels already given which $\mathcal{A}$ will always assign). First, $v_1, \ldots, v_{n-2}$ form a path where every even-indexed edge has label $T_{\max}$, Second, $v_{n-1}$ has an edge to each $v_1, \ldots v_{n-2}$ with fixed label $T_{\max}-2$, $v_{n-1}$ and $v_n$ share an edge with fixed label $T_{\max}-1$, and $v_n$ has an edge to each $v_1, \ldots v_{n-2}$ with fixed label $T_{\max}$.

Assume $\mathcal{D}$ acts arbitrarily. For edges with fixed labels, $\mathcal{A}$ responds that label. For all other edges, $\mathcal{A}$ replies "infection failed" as long as at least one consistent label would remain; otherwise, it replies with any consistent label.

We apply the potential argument from Theorem 4 to all edges that do not have a fixed label. We call those $(n-3)/2$ edges *relevant*. Initially, each relevant edge has $T_{\max}$ possible labels, so $\Phi(0) \geq (n-3)/2 \cdot T_{\max}$. In each round, for every node that gets infected, at most $\delta$ labels are removed per adjacent relevant edge, either due to failed infection attempts, or a successful infection with $\delta$ or fewer labels remaining. Thus, the potential decreases by at most $\delta k$ per round. Dividing the initial potential by this maximum decrease shows that any $\mathcal{D}$ needs at least $\lfloor (n \cdot (T_{\max}-3))/(2\delta k) \rfloor$ rounds. □

### 5.1 Witness Complexity

The witness complexity of a temporal graph is the minimum number of rounds required for a $\mathcal{D}$, knowing all labels, to convince an observer of the labeling's correctness.

**Definition 4.** *A length $a$* witnessing schedule *for a temporal graph $\mathcal{G}$ is a sequence of seed infection sets $S_1, \ldots, S_a$ such that after performing $a$ rounds with the respective seed infection sets, all labels in the graph are uniquely determined by the logs of these rounds. The* witness complexity *of a temporal graph $\mathcal{G}$ is the length of its shortest witnessing schedule.*

Observe that the graphs constructed in the proof of Theorem 5 have witness complexity $O(n)$, which is significantly lower than their graph discovery complexity. However, witness complexity is a powerful tool for establishing lower bounds on graph discovery complexity, particularly when seeking bounds independent of $T_{\max}$. The following lemma states the formal relationship between the two complexities.

⋆ **Lemma 2.** *Let $\mathcal{G}$ be a temporal graph and $T_{\max}$, $\delta$, $k$ parameters as defined above. Then the witness complexity of $\mathcal{G}$ in this instance is at most as large as the graph discovery complexity for the same parameters.*

Note however, that the witness complexity technique can only ever be at most the number of edges in a graph.

⋆ **Theorem 6.** *For any instance $(\mathcal{G}, T_{\max}, \delta, k)$, the witness complexity is at most $|E(\mathcal{G})|$.*

We now show that this worst case is actually tight, and there are graphs that asymptotically require about one round per edge to witness correctly. Observe that this bound is irrespective of $k$, thus does not have one of the major shortfalls of our previous lower bounds for the TGD game.

⋆ **Theorem 7.** *There is an infinite family of temporal graphs $\{\mathcal{G}_n\}_{n\in\mathbb{N}}$ whose witness complexity grows in $\Omega_k(|E(\mathcal{G}_n)|)$.*

We will now define $\{\mathcal{G}_n\}_{n\in\mathbb{N}}$. Then we formalize when an infection attempt is *relevant*, in the sense that it makes meaningful progress towards winning the witness complexity game. Lastly, we show that there can be at most one such relevant infection attempt per round in $\{\mathcal{G}_n\}_{n\in\mathbb{N}}$.

Intuitively, each graph in $\{\mathcal{G}_n\}_{n\in\mathbb{N}}$ contains four node sets: $L$, $R$, $B$, and $C$. The nodes in $L$ and $R$ form a complete bipartite graph, where the edges between a node in $L$ and all nodes in $R$ are assigned to distinct *phases*. The nodes in $R$ are connected such that once a node in $R$ is infected, it spreads the infection through $R$ without additional input from $L$. The sets $B$ and $C$ serve as gadgets to ensure that at the end of each phase, all nodes in $L$ and $R$ become infected via edges outside $L$ and $R$. This prevents further information being gathered about the labels between $L$ and $R$, ensuring the lower bound on the complexity of the discovery process.

For $x \in \mathbb{N}$, we construct a temporal graph $\mathcal{G}_n$ of size $n := 5x$. The vertices are given by $V(\mathcal{G}_n) := L \sqcup R \sqcup B \sqcup C$ with $L := \{\ell_1, \dots, \ell_x\}, R := \{r_1, \dots, r_{2x}\}, B := \{b_1, \dots, b_x\}, C := \{c_1, \dots, c_x\}$. Denote by $R_2$ the nodes in $R$ with an even index. Then the edges are given by $E(\mathcal{G}_n) := L \times R_2 \sqcup R^2 \sqcup L \times B \sqcup B \times R \sqcup B^2 \sqcup C \times R \sqcup \{b_i c_i \mid i \in [x]\}$. The labels on the edges are defined as follows:

Let $P = \{p_1, \dots, p_x\}$ be a set of $x$ edge-disjoint Hamiltonian paths on $R$. Such paths exists by [Axiotis and Fotakis, 2016]. Note that by their construction, for every path, there are at most two nodes with an odd index in a row. Assume without loss of generality that each $p_j$ begins at $r_{2j}$, and write $r_{p(i,j)}$ for the $j$-th node from $R_2$ on the path $p_i$. Denote by $p^{-1}(i,j)$ its index on $p_i$. Let $k$ be arbitrary, $T_{\max} := x(4x+1)$, and $\delta := 4x+1$. Now,

$$
\begin{aligned}
&\text{for } \ell_i \in L, j \in [x], \text{ set} && \lambda(\ell_i, r_{p(i,j)}) := \\
&&& \quad i\delta + 4(p^{-1}(i,j)) + 1, \\
&\text{for } p_q \in P, j \in [x], \text{ set} && \lambda((p_q)_j, (p_q)_{j+1}) := q\delta + 2j + 1, \\
&\text{for } b_i \in B, j \in [x], \text{ set} && \lambda(b_i, r_{(p_i)_j}) := i\delta + 2j + 2, \\
&\text{for } i < j \in [x], \text{ set} && \lambda(b_i, b_j) := (i+1)\cdot\delta - 2, \\
&\text{for } i \in [x], \text{ set} && \lambda(b_i, c_i) := (i+1)\cdot\delta - 1, \\
&\text{for } i > j \in [x], \text{ set} && \lambda(b_i, c_j) := (i+1)\cdot\delta - 2, \\
&\text{for } c_i \in C, r_j \in R, \text{ set} && \lambda(c_i, r_j) := (i+1)\cdot\delta, \\
&\text{for } \ell_i \in L, b_j \in B, \text{ set} && \lambda(\ell_i, b_j) := (j+1)\cdot\delta.
\end{aligned}
$$

Observe that the temporal edges of $\mathcal{G}_n$ can be partitioned into $x$ sets, each having labels in a fixed interval of size $\delta$ and for $i \in [x]$, let $E_i = \{e \in E(\mathcal{G}_n) \mid \lambda(e) \in [\delta \cdot i, \delta \cdot (i+1) - 1]\}$. We refer to the edges $E_i$ and corresponding intervals as *phases*. Now, an infection attempt between some $l_i \in L$ and $r_{2j} \in R_2$ is called *relevant* if (i) it happens at $\lambda(l_i r_{2j})$ and is successful, or (ii) it happens at $\lambda(l_i r_{2j}) - 1$, is unsuccessful, and exactly one endpoint was infected before the attempt.

The result then follows from the following three properties: (1) for each edge in $L \times R_2$, there has to be at least one relevant infection attempt to win the witness complexity game, (2) there is at most one relevant infection per phase, and (3) there is at most one phase with a relevant infection per round.

| Lower Bound | Upper Bound |
|---|---|
| **Basic model** | |
| $\Omega_k(m)$ | $\mathcal{O}_k(m + \lvert EC_\delta(\mathcal{G})\rvert\, T_{\max}/\delta)$ |
| **Infection times only** | |
| $\Omega_k(m)$ | $\mathcal{O}_k(m + \lvert EC_\delta(\mathcal{G})\rvert\, T_{\max}/\delta)$ |
| **Unknown static graph** | |
| $\Omega_{m,\lvert EC_\delta(\mathcal{G})\rvert}(nT_{\max}/(\delta k))$ | $\mathcal{O}(nT_{\max})$ |
| **Multilabels** | |
| $\Omega_{\lvert EC_\delta(\mathcal{G})\rvert}(nT_{\max}/(\delta k))$ | $\mathcal{O}(nT_{\max})$ |
| **Multiedges** | |
| $\Omega(nT_{\max}/(\delta k))$ | $\mathcal{O}_k(m + \lvert EC_\delta(\mathcal{G})\rvert\, T_{\max}/\delta)$ |

Table 1: Overview of upper and lower bounds on the number of rounds for different variations of the graph discovery game. A subscript to a Landau symbol indicates variables that the asymptotic growth is independent of.

## 6 Extending the Model

So far, we examined a simple version of the TGD game with restricted assumptions about infection behavior and the knowledge of $\mathcal{D}$. These might seem restrictive and less close to the real-world processes. In this section, we lift these restrictions and show that either the theoretical behavior remains unchanged, or the problem becomes trivial, offering no significant improvement over brute-force. See Table 1 for an overview of the resulting lower bounds and algorithms.

In summary, our `DiscoveryFollow` algorithm works if $\mathcal{D}$ only gets infection-time feedback or if we allow multiedges, lifting the two most restrictive assumptions previously made. We also prove that variations where $\mathcal{D}$ needs to ensure the non-existence of a high number of edges or labels (such as the unknown static graph or multilabel variations) do not allow for significant improvements over the trivial algorithm and that our `DiscoveryFollow` algorithm is not applicable. Results and proofs of this section have been moved to the appendix due to space constraints.

## 7 Experimental Evaluation

The gap between the lower and upper bounds for the TGD problem is small, but only tells us about the worst-case performance, motivating us to investigate the performance of our TGD algorithm on common synthetic graphs and real-world data. We formulate three hypotheses we aim to test.

**Hypothesis 1.** *The number of rounds required to discover a temporal graph is linear in the number of edge labels.*

This first hypothesis is motivated by the worst-case analysis from Theorem 3. We aim to test how closely the performance of our algorithm in practice matches this theoretical bound. Note, this also takes into account the effect of the additional optimization described in the setup.

**Hypothesis 2.** *Graphs with higher density spend fewer rounds on component discovery.*

The `DiscoveryFollow` algorithm works in two distinct phases: (1) the main loop in Algorithm 2 discovers new $\delta$-ecc's (the *component discovery phase*) and (2) the `Explore`

routine explores the found components (the *component exploration phase*). Both of these stages require $\mathcal{D}$ to spend rounds, and their respective cost is dependent on the structure of the graph to be explored. As we prove in Theorem 3, the component discovery phase is triggered at most $|EC_\delta(\mathcal{G})|$ times. We hypothesize that $|EC_\delta(\mathcal{G})|$ tends to be lower in denser graphs as the components tend to merge as more edges are inserted, which would lead to a relatively lower cost for component discovery as compared to component exploration.

**Hypothesis 3.** *In Erdős-Renyi graphs, the percentage of rounds spend on component discovery shows a threshold behavior in $T_{\max}/(|V(\mathcal{G})| \cdot p)$. This is mediated by $|EC_\delta(\mathcal{G})|$.*

Hypothesis 3 is a specification of Hypothesis 2 for Erdős-Renyi graphs. We can view this hypothesis as the temporal extension of the typical threshold behavior that static Erdős-Renyi graphs exhibit in $np$ [Erdos *et al.*, 1960]. We also conjecture this behavior holds provably in Conjecture 1.

**Setup.** We perform our evaluation on two data sets. One is synthetically generated, while the other is real-world data. First, we evaluate our algorithm on Erdős-Renyi graphs. While the model has been developed for static graphs, it is commonly extended to temporal graphs [Angel *et al.*, 2020]. To generate a simple temporal graph from an Erdős-Renyi graph, we simply choose each edge label uniformly at random from the set $[T_{\max}]$. This means that $T_{\max}$ is now the third parameter to generate these graphs, in addition to the classical ones $n$ (the number of nodes) and $p$ (the density parameter). Denote such a graph as $ERT(n, p, T_{\max})$.
Secondly, to evaluate our algorithm on real-world data, we employ a data set from the Stanford Large Network Dataset Collection [Kumar *et al.*, 2021]. The `comm-f2f-Resistance` collection is described by the project as a set of 62 "dynamic face-to-face interaction network[s] between a group of participants".

In our implementation, we employ a small optimization upon Algorithm 2. We skip what we call *redundant seed infections*. A seed infection $(v, t) \in V(\mathcal{G}) \times [0, T_{\max}]$ is *redundant* if we already know the labels of all edges adjacent to $v$ at the time in the algorithm we would perform this seed infection. This does not depend on $t$. For an illustration, consider Figure 2. There, the `DiscoveryFollow` algorithm would, after discovering the only edge adjacent to the leftmost node by a seed from its other adjacent node, perform seed infections at the leftmost node even though we already know the label of the only edge we could possibly discover.

We run `DiscoveryFollow` on all graphs from the `comm-f2f-Resistance` dataset and for a wide range of parameters for the temporal Erdős-Renyi graphs. We test with 1 to 100 nodes in steps of size 5, for probabilities $p \in \{.01, .05, .1, .15, .2, .25, .3, .35, .4, .5, .7, .9\}$. We pick $T_{\max}$ as a factor of $n$, testing with $T_{\max}/n \in \{.05, .1, .2, .3, .4, .5, .7, .9, 1, 2, 3, 5, 7, 10\}$. Similarly, $\delta$ is 1 or a multiple of $T_{\max}$, namely $\delta/T_{\max} \in \{.01, .05, .1, .3, .5\}$ We record the number of rounds needed to complete the discovery and how many of these rounds are spent in the component discovery versus the component exploration phases of the algorithm. Both our implementation and analysis code are available under a permissive open-source license and can be used to replicate our findings.[1]

### 7.1 Results

We now critically evaluate our hypotheses against the data thus obtained and compare the effects between the different data sets and parameters. Particularly, we pay attention to evidence on how the different hypotheses interact. Finally, we give Conjecture 1 as a result of our analysis of Hypothesis 3.

**Hypothesis 1.** To investigate our first hypothesis, we compare how the number of rounds relates to the number of edges. See Fig. 3 for the results on our datasets. We can see that in Erdős-Renyi graphs, the relationship closely follows the $6\,|E(\mathcal{G})|$ line predicted by our theoretical results. Clearly, this effect is influenced by other parameters such as $p$, $T_{\max}$, and $n$, whose roles we will examine in the discussion of the other hypotheses. However, if we consider graphs with the same $p$ and ratio $T_{\max}/n$ (i.e., one facet and one color in Fig. 3), we see that the relationship is strictly linear—the points form a tightly distributed straight line. We see that the trend for $p \leq 0.25$ is slightly above $6\,|E(\mathcal{G})|$ and slightly under it for larger values of $p$. This occurs since, when $p$ is small, more time is spent on component discovery than on component exploration. We will explore this more thoroughly in the discussion of the results regarding Hypothesis 2.

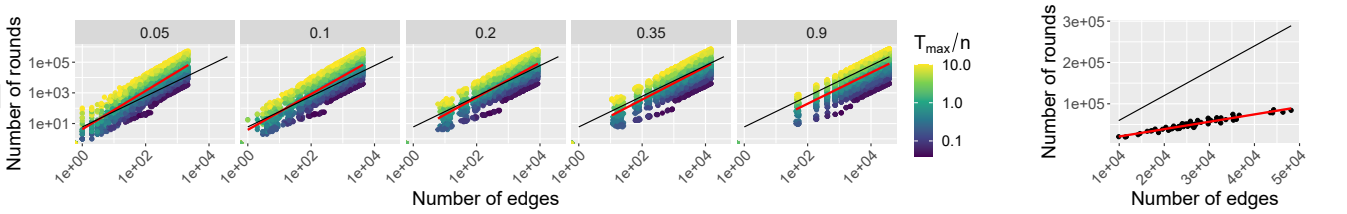
In the SNAP data set, the trend is linear in the number of edges, but significantly less than $6\,|E(\mathcal{G})|$ rounds are required to discover the graph. In fact, the gradient of the regression line is only 1.78. This can be explained by the optimization skipping redundant infections as outlined in our setup, as this enables the algorithm to require less than the 6 infections per edge, which would otherwise be strictly required. In summary, while there is a strong linear relationship following the $6\,|E(\mathcal{G})|$ line, the number of rounds also significantly depends on other properties of the graph. We can see these effects in both synthetic and real-world data.

**Hypothesis 2.** As this hypothesis is about the relationship between graph density and time spent on component discovery, we only analyze it on the Erdős-Renyi graphs. The SNAP dataset does not have significant differences in graph density.

In Figure 4a, we plot the percentage of time spent on component discovery dependent on the parameter $p$ (which specifies the density of the graph). We observe a clear and strong, inversely proportional relationship. This leads us to accept Hypothesis 2. This is explained from the theoretical analysis of `DiscoveryFollow`, as we expect denser graphs to have fewer $\delta$-ecc's, thus less need to discover new components.

**Hypothesis 3.** Examining Fig. 4b, we see a threshold between $np/T_{\max} = 0.01$ and $np/T_{\max} = 1.00$ where we move from spending only a small amount of rounds on component discovery to spending close to all of our rounds on component discovery. Our hypothesis, that this is mediated by the size of the $\delta$-ecc's, is supported by the results in Fig. 4c. The additional differentiation by color for larger values of $np/T_{\max}$ can be explained since the size of a component is constrained by the size of the graph.

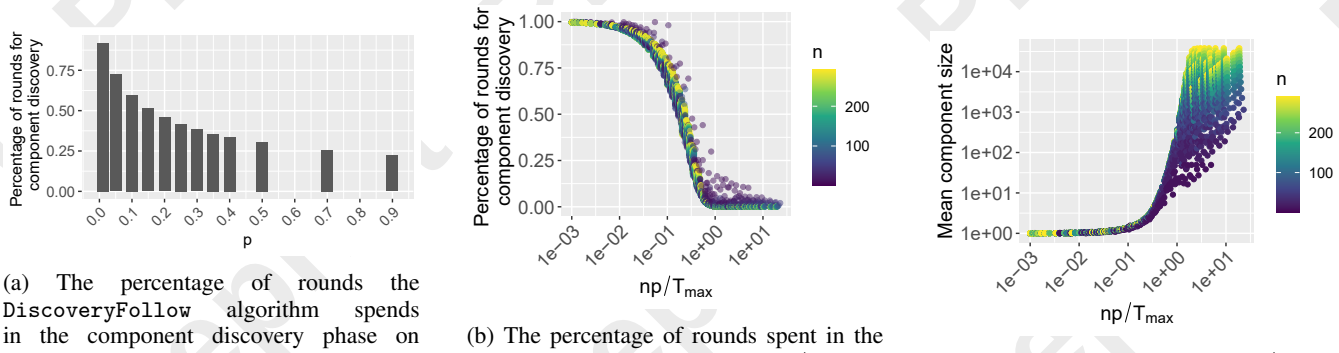
[1] See https://github.com/BenBals/dynamic-network-discovery.




(a) Erdős-Renyi graphs for different densities $p$

(b) SNAP dataset

Figure 3: Number of rounds by number of edges for the `DiscoveryFollow` algorithm on the two data sets. The black line indicates the $6\,|E(\mathcal{G})|$ line (our theoretical bound for the component exploration phase), and the red line is the trend line (i.e., the linear regression).




(a) The percentage of rounds the `DiscoveryFollow` algorithm spends in the component discovery phase on temporal Erdős-Renyi graphs by density $p$.

(b) The percentage of rounds spent in the component discovery phase by $np/T_{\max}$.

(c) Mean size of the $\delta$-ecc's by $np/T_{\max}$.

Figure 4: Cost of component discovery in the `DiscoveryFollow` algorithm. Subfigures (b) and (c) use logarithmic axes, and the color denotes the number of nodes.

Note that big $\delta$-ecc's imply that we can discover many edges in the component exploration phase for a single node explored in the component discovery phase. That means the plotted percentage shrinks as the average size of $\delta$-ecc's increases. This behavior is similar to that seen in static Erdős-Renyi graphs, where if $np = 1$, there almost surely is a largest connected component (in the classical sense) of order $n^{2/3}$. And if $np$ approaches a constant larger than 1, there asymptotically almost surely is a so-called giant component containing linearly many nodes [Erdos *et al.*, 1960]. This motivates us to conjecture the following similar behavior.

**Conjecture 1.** *Let $\delta = 1, p \in [0,1], T_{\max}, n \in \mathbb{N}$ and $\mathcal{G} := ERT(n,p,T_{\max})$. Then*

- *if $T_{\max}/(np) \xrightarrow{n\to\infty} c$ where $c < 1$, then $|EC_\delta(\mathcal{G})| \,/\, |E(\mathcal{G})| \xrightarrow{n\to\infty} 0$, and*
- *if $T_{\max}/(np) > 1$, then almost surely, all $\delta$-ecc's have size at most $\mathcal{O}(\log n^2) = \mathcal{O}(\log n)$.*

Investigating the connectivity behavior of random temporal graphs in a way that respects their inherent temporal aspects is an interesting and little understood problem [Casteigts *et al.*, 2022; Casteigts *et al.*, 2024]. The authors theoretically study sharp thresholds for connectivity in temporal Erdős-Renyi graphs, and conjecture about a threshold for the emergence of a node-based giant (i.e., linear size) connected component. This conjecture can be seen as capturing the equivalent behavior under waiting time constraints (i.e., where edges are only considered if their edge labels do not differ too much).

## 8 Conclusion

We give a comprehensive theoretical and empirical study of TGD in temporal graphs. `DiscoveryFollow` provides an efficient TGD strategy requiring only $6\,|E(\mathcal{G})| + |EC_\delta(\mathcal{G})|\,\lceil T_{\max}/\delta \rceil$ rounds. Our lower bound proves that any algorithm must spend at least $\Omega(|E(\mathcal{G})|)$ rounds, showing `DiscoveryFollow` is close to optimal. Our empirical analysis highlights the relevance of our theoretical results for practical applications and gives rise to interesting insights of its own. We see that on Erdős-Renyi graphs, the observed performance of `DiscoveryFollow` matched our theoretical analysis. On real-world data from the SNAP collection, the algorithm even slightly outperforms our predictions. Finally, we observe a close link between the parameters of the temporal Erdős-Renyi model, the temporal connectivity structure of the resulting graphs, and our algorithmic performance, creating a bridge between our theoretical insights on $\delta$-connected components and their empirical behavior.

Future work can tighten the lower bound from Theorem 5 and give a lower bound that is tight in $k$ and $\delta$. Another avenue is to investigate $|EC_\delta(\mathcal{G})|$ further. In particular, to prove or disprove the observed threshold in Erdős-Renyi graphs (Conjecture 1) and investigate an analogue to the static $\ln n/n$ threshold. Finally, future research can explore more variations of TGD, such as finding specific nodes or checking for structural properties instead of discovering the whole graph.

## References

[Akrida *et al.*, 2019] Eleni C Akrida, Jurek Czyzowicz, Leszek Gasieniec, Łukasz Kuszner, and Paul G Spirakis. Temporal flows in temporal networks. *Journal of Computer and System Sciences*, 103:46–60, 2019.

[Angel *et al.*, 2020] Omer Angel, Asaf Ferber, Benny Sudakov, and Vincent Tassion. Long monotone trails in random edge-labellings of random graphs. *Combinatorics, Probability and Computing*, 29(1):22–30, 2020.

[Axiotis and Fotakis, 2016] Kyriakos Axiotis and Dimitris Fotakis. On the Size and the Approximability of Minimum Temporally Connected Subgraphs. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55, pages 149:1–149:14, 2016.

[Berenbrink *et al.*, 2023] Petra Berenbrink, Max Hahn-Klimroth, Dominik Kaaser, Lena Krieg, and Malin Rau. Inference of a rumor's source in the independent cascade model. In *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, volume 216 of *Proceedings of Machine Learning Research*, pages 152–162. PMLR, 31 Jul–04 Aug 2023.

[Casteigts *et al.*, 2021a] Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, and Philipp Zschoche. Finding temporal paths under waiting time constraints. *Algorithmica*, 83:2754 – 2802, 2021.

[Casteigts *et al.*, 2021b] Arnaud Casteigts, Kitty Meeks, George B. Mertzios, and Rolf Niedermeier. Temporal Graphs: Structure, Algorithms, Applications (Dagstuhl Seminar 21171). *Dagstuhl Reports*, 11(3):16–46, 2021.

[Casteigts *et al.*, 2022] Arnaud Casteigts, Michael Raskin, Malte Renken, and Viktor Zamaraev. Sharp thresholds in random simple temporal graphs. In *Symposium on Foundations of Computer Science (FOCS 2022)*, pages 319–326, 2022.

[Casteigts *et al.*, 2024] Arnaud Casteigts, Timothée Corsini, and Writika Sarkar. Simple, strict, proper, happy: A study of reachability in temporal graphs. *Theoretical Computer Science*, 991:114434, 2024.

[Chistikov *et al.*, 2024] Dmitry Chistikov, Luisa Estrada, Paolo Turrini, and Mike Paterson. Learning a social network by influencing opinions. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024)*. AAMAS, 2024.

[Danda *et al.*, 2021] Umesh Sandeep Danda, G. Ramakrishna, Jens M. Schmidt, and M. Srikanth. On short fastest paths in temporal graphs. In Ryuhei Uehara, Seok-Hee Hong, and Subhas C. Nandy, editors, *WALCOM: Algorithms and Computation*, pages 40–51, Cham, 2021. Springer International Publishing.

[Daneshmand *et al.*, 2014] Hadi Daneshmand, Manuel Gomez-Rodriguez, Le Song, and Bernhard Schoelkopf. Estimating diffusion network structures: Recovery conditions, sample complexity & soft-thresholding algorithm. In *International conference on machine learning*, pages 793–801. PMLR, 2014.

[Deligkas *et al.*, 2024] Argyrios Deligkas, Michelle Döring, Eduard Eiben, Tiger-Lily Goldsmith, and George Skretas. Being an influencer is hard: The complexity of influence maximization in temporal graphs with a fixed source. *Information and Computation*, page 105171, 2024.

[Erdos *et al.*, 1960] Paul Erdos, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 5(1):17–60, 1960.

[Gayraud *et al.*, 2015] Nathalie T.H. Gayraud, Evaggelia Pitoura, and Panayiotis Tsaparas. Diffusion maximization in evolving social networks. In *Proceedings of the 2015 ACM on Conference on Online Social Networks*, COSN '15, page 125–135, New York, NY, USA, 2015. Association for Computing Machinery.

[Hethcote, 1989] Herbert W. Hethcote. *Three Basic Epidemiological Models*, pages 119–144. Springer Berlin Heidelberg, Berlin, Heidelberg, 1989.

[Kempe *et al.*, 2003] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 137–146, 2003.

[Kermack *et al.*, 1927] William Ogilvy Kermack, A. G. McKendrick, and Gilbert Thomas Walker. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 115(772):700–721, 1927.

[Kumar *et al.*, 2021] Srijan Kumar, Chongyang Bai, VS Subrahmanian, and Jure Leskovec. Deception detection in group video conversations using dynamic interaction networks. In *ICWSM 2021*. International AAAI Conference on Web and Social Media, 2021.

[Leskovec *et al.*, 2007] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, page 420–429, New York, NY, USA, 2007. Association for Computing Machinery.

[Lokhov, 2016] Andrey Lokhov. Reconstructing parameters of spreading models from partial observations. *Advances in Neural Information Processing Systems*, 29, 2016.

[Luo *et al.*, 2013] Wuqiong Luo, Wee Peng Tay, and Mei Leng. Identifying infection sources and regions in large networks. *IEEE Transactions on Signal Processing*, 61(11):2850–2865, 2013.

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

[Netrapalli and Sanghavi, 2012] Praneeth Netrapalli and Sujay Sanghavi. Learning the graph of epidemic cascades. *ACM SIGMETRICS Performance Evaluation Review*, 40(1):211–222, 2012.

[Park and Honorio, 2016] Keehwan Park and Jean Honorio. Information-theoretic lower bounds for recovery of diffusion network structures. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 1346–1350. IEEE, 2016.

[Pastor-Satorras *et al.*, 2015] Romualdo Pastor-Satorras, Claudio Castellano, Piet Van Mieghem, and Alessandro Vespignani. Epidemic processes in complex networks. *Reviews of modern physics*, 87(3):925–979, 2015.

[Pouget-Abadie and Horel, 2015] Jean Pouget-Abadie and Thibaut Horel. Inferring graphs from cascades: A sparse recovery framework. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, page 625–626, New York, NY, USA, 2015. Association for Computing Machinery.

[Rong *et al.*, 2016] Yu Rong, Qiankun Zhu, and Hong Cheng. A model-free approach to infer the diffusion network from event cascade. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, page 1653–1662, New York, NY, USA, 2016. Association for Computing Machinery.

[Zhu and Ying, 2016] Kai Zhu and Lei Ying. Information source detection in the sir model: A sample-path-based approach. *IEEE/ACM Transactions on Networking*, 24(1):408–421, 2016.