Catch Me If You Can: Finding the Source of Infections in Temporal Networks

Ben Bals
CWI and Vrije Universiteit
Amsterdam, The Netherlands
bjjb@cwi.nl

Michelle Döring
Hasso Plattner Institute
Potsdam, Germany
michelle.doering@hpi.de

Nicolas Klodt
Hasso Plattner Institute
Potsdam, Germany
nicolas.klodt@hpi.de

George Skretas
Hasso Plattner Institute
Potsdam, Germany
georgios.skretas@hpi.de

ABSTRACT

Source detection (SD) is the task of finding the origin of a spreading process in a network. Algorithms for SD help us combat diseases, misinformation, pollution, and more, and have been studied by physicians, physicists, sociologists, and computer scientists. The field has received considerable attention and been analyzed in many settings (e.g., under different models of spreading processes), yet all previous work shares the same assumption that the network the spreading process takes place in has the same structure at every point in time. For example, if we consider how a disease spreads through a population, it is unrealistic to assume that two people can either never or at every time infect each other, rather such an infection is possible precisely when they meet. Therefore, we propose an extended model of SD based on temporal graphs, where each link between two nodes is only present at some time step. Temporal graphs have become a standard model of time-varying graphs, and, recently, researchers have begun to study infection problems (such as influence maximization) on temporal graphs [8, 11]. We give one of the first formalizations of SD on temporal graphs. For this, we employ the standard SIR model of spreading processes [12]. We give both lower bounds and algorithms for the SD problem in a number of different settings, such as with consistent or dynamic source behavior and on general graphs as well as on trees.

KEYWORDS

Temporal Graphs; Infection Models; Source Detection

ACM Reference Format:
Ben Bals, Michelle Döring, Nicolas Klodt, and George Skretas. 2026. Catch Me If You Can: Finding the Source of Infections in Temporal Networks. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 8 pages. https://doi.org/10.65109/LOWK9942

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/LOWK9942

1 INTRODUCTION

Information, diseases, pollutants—all these things spread through the networks that define how everything from people and servers to our streets and sewers are connected. The field of spreading processes has received substantial attention from researchers as diverse as physicians, physicists, sociologists, and computer scientists. One of the central questions that naturally arises is to find the source of some spreading process given only limited information (e.g., from a small set of disease infection times or from a number of sensors in a sewer network) [19, 21, 24, 26]. The current mainstream models in this area, such as the independent cascade model [16], or the susceptible-infected-resistant (SIR) model [12, 17, 24] incorporate the inherent temporal behavior of the spreading process, that is, infections themselves take time such that a node first gets infected and only then is able to infect other nodes. However, they fail to capture temporal dependencies in the underlying network itself. In many real-world networks the connections change over time (e.g., social networks in which diseases or information spread are highly temporal as the link between two people is only able to transmit information or diseases at specific points in time) and these changes heavily impact the behavior of spreading processes. For this reason, the study of infections in temporal graphs has been initiated. In 2015, Gayraud et al. [11], generalized the independent model from static to temporal graphs. Since then, very few people have studied problems related to spreading processes in temporal networks. One of the most fundamental problem in spreading processes is the *source detection* (SD) problem. We provide a rigorous definition of SD in temporal graphs and give both algorithms and lower bounds for this problem in a variety of settings.

Our contribution is twofold: (i) we define the SD problem as a round-based, interactive two-player game, (ii) we provide algorithms and lower bounds for different parameters of the SD game, and we provide theoretical proofs for all of our claims. See Table 1 for an overview of our results. *Statements where proofs or details are omitted due to space constraints are marked with ⋆. A full version of this paper is available at https://arxiv.org/abs/2412.10877.*

In Section 3, we formalize the SD problem as a round-based game in which two players interact with each other. The *Adversary* decides the structure of the underlying temporal network, as well as which node is the source and when it starts an infection chain. The *Discoverer* initially has no information about the location, number, or label of the edges. In each round the infection process plays out

Table 1: An overview of the price of detection in different settings. For lower bounds, a *wcp* annotation signifies that the bound holds for all Discoverer algorithms winning the game with constant probability. For upper bounds, *wcp* signifies that a Discoverer algorithm winning the game with constant probability and the noted price of detection exists. Upper bounds marked *det* are achieved via a deterministic algorithm. Results marked [a] are transferred between known and unknown settings (i.e., lower bounds from known to unknown and upper bounds in the other direction). Similarly, results transferred between the consistent and obliviously dynamic settings are marked [b] and transfers between trees and general graphs are marked [c]. The result marked * only holds when the Discoverer is allowed to watch two nodes.

	Trees		General	
	Lower Bound	Upper Bound	Lower Bound	Upper Bound
Consistent				
Known	$\Omega(n \log n)$ wcp (Thm. 5)	$O(n \log n)$ wcp (Cor. 1)	$\Omega(n\sqrt{n})$ wcp (Thm. 3)	$O(n\sqrt{n})$ wcp[a]
Unknown	$\Omega(n\sqrt{n})$ wcp (Thm. 2)	$O(n\sqrt{n})$ wcp[c]	$\Omega(n\sqrt{n})$ wcp[c]	$O(n\sqrt{n})$ wcp (Thm. 1)
Obliviously dynamic				
Known	$\Omega(n \log n)$ wcp[b]	$O(n \log n)$ wcp* (Thm. 8)	$\Omega(n^2)$ wcp (Thm. 7)	n^2 det[a]
Unknown			$\Omega(n^2)$ wcp[a]	n^2 det (Thm. 6)

once and the Discoverer may watch a single node and is informed if, when, and by which neighbor this watched node is infected. Note that the Adversary does not know which nodes the Discoverer watches. We study both the setting where the infections in each round will be identical (*consistent* source behavior) or where the Adversary may vary the time step the infection starts but not its source node (*obliviously dynamic* source behavior). Counting the rounds of this game until the Discoverer has found the source is an insufficient cost metric, as then the Adversary is incentivized to minimize infections in order to evade detection. Instead, we study the number of infections until detection, which we call the *price of detection*. This cost measure combines an incentive to avoid detection (to prolong the period in which the Adversary may perform infections) and an incentive to infect as many nodes as possible in a single round. Notice, that if n is the number of nodes in the graph, any Discoverer algorithm must always tolerate $\Omega(n^2)$ infections in the worst case, and there is an algorithm that always wins the game within n^2 infections (see Theorem 6).

Because in this game, the worst-case performance of any algorithm is trivially bad (i.e., its price of detection is in $\Omega(n^2)$), we explore randomized techniques to overcome this limitation in Section 4. In particular, we give a Discoverer algorithm that wins the SD game with constant probability (i.e., a probability that does not decrease for larger networks) while only tolerating $O(n\sqrt{n})$ infections. We also prove that this price of detection is asymptotically optimal for any algorithm that wins the game with a constant probability.

In Section 5, we explore how the problem changes if the Discoverer has knowledge about the underlying static graph (but not about when an edge exists in the temporal graph). Surprisingly, we prove that this does not lead to a decreased price of detection. In particular, we show that the lower bounds from the unknown setting transfer to the known one. Despite this, we can show positive results on structured instances. We give an algorithm that, for graphs with bounded treewidth tw, wins the game on known static graphs with constant probability with a price of detection of $O(\text{tw} \cdot n \log n)$. This directly translates to a $O(n \log n)$ algorithm on trees, which we discuss in Section 6. Treewidth is a structural graph parameter that, in a specific technical sense, captures how close a graph is to a tree. Throughout the paper we will see problem variations that become significantly easier on trees. The treewidth parameter helps us extend these results, in some sense showing that some of the difficulty of these settings directly relates to how tree-like a graph is (instead of only examining the binary distinction tree vs. not a tree).

In Section 7, we study what happens when we allow the Adversary to change the time-step at which it infects the source between each round. Note that we still assume the Adversary does not know which nodes the Discoverer chooses to watch. That is, while the source behavior may be dynamic, it may not depend on the actions of the Discoverer, thus, we call this model *obliviously dynamic*. We first show that any Discoverer algorithm that wins the game with constant probability must have a price of detection in $\Omega(n^2)$. On a more positive note, we show that if we strengthen the Discoverer slightly by allowing them to watch two nodes each round, we are still able to achieve a price of detection of $O(n \log n)$ with constant probability on trees (if the static graph is known). We finish this off by giving a proof that allowing the Discoverer to watch k nodes may only decrease the price of detection by at most a factor of k if the source behavior is consistent, thus this generalization may only be asymptotically different in the obliviously dynamic setting.

Related Work. The SIR model and related models are standard in mathematical biology [12] and have been extensively studied from a statistical perspective [3]. The SIR model most closely models how viral infections spread through populations. For example, adaptations of the model have been used to study COVID-19 [5, 6, 18].

Recently, researchers have started to ask more algorithmic questions about infection models, such as SD in addition to the classical task of predicting spread. In particular, [16] set off this new wave of algorithmic study of infections. They introduce and study the *influence maximization problem* (though under the independent cascade model instead of SIR), which already incorporates the inherently temporal nature of spreading infections while the underlying graph

is modeled as static. Their work finds broad resonance with applications as diverse as the study of misinformation [4, 25], infectious diseases [5, 9], and viral marketing [10, 20].

The SD problem has a rich literature and has even been studied on the SIR model we employ. In 2011, [24] study the SD problem under the SIR model. In 2016, [26] extend their work. Other notable algorithmic contributions include the PVTA algorithm proposed by [22], which focuses on estimating the source of an infection from observations on a sparse set of nodes. This model has also recently been extended to SD [2]. In 2011, [24] study the SD problem under the SI model. [26] extend this work to the SIR model. [11], extend the study of influence maximization on both the independent cascade and linear threshold models to temporal graphs (which were popularized by [15]). [8] study a number of variations of the influence maximization problem on temporal graphs under the SIS model (which is closely related to SIR).

Most closely to our approach, Huang [13] studied the problem of detecting a source in a known temporal network. Their approach uses a backwards diffusion process based on the assumption that infections are more likely to travel via short (or shortest) paths. [23] extend this work significantly, both by studying more general settings and by providing algorithmic tools. Both papers differ from our framework in one main assumption: that the temporal graph is known to the Discoverer. This is not realistic in many practical settings and we relax this assumption. These works also differ in that the discoverer is non-interactive, that is, it cannot influence the infection data collected based on the information received so far.

2 PRELIMINARIES

We write $2\mathbb{N}$ for the set of even natural numbers and $2\mathbb{N}+1$ for the set of odd natural numbers. Two real-valued functions $f, g\colon \mathbb{N} \to \mathbb{R}$ are asymptotically equivalent if $(f(n)/g(n)) \xrightarrow{n\to\infty} 1$. We use the standard definition of *treewidth* as presented in [7].

We define a *temporal graph* $\mathcal{G} = (V, E, \lambda)$ with lifetime $T_{\max}$ as an undirected graph (V, E) together with a *labeling function* $\lambda\colon E \to 2^{[T_{\max}]}$. We interpret this as $e \in E$ being present precisely at time steps $\lambda(e)$. We restrict ourselves to *simple* temporal graphs where each edge has exactly one label. Abusing notation, we therefore also use λ as if it were defined as $\lambda\colon E \to [T_{\max}]$. We also write $V(\mathcal{G})$ for the nodes of $\mathcal{G}$, and $E(\mathcal{G})$ for its edges. Every temporal graph has an *(underlying) static graph* $G = (V, E)$ obtained by simply forgetting the edge labels. A sequence of nodes $v_1, \ldots, v_\ell \in V$ is a *temporal path* if it forms a path in G and the labels along the edges are strictly increasing (i. e., for all $i \in [\ell - 2]$, we have $\lambda(v_i v_{i+1}) < \lambda(v_{i+1} v_{i+2})$).

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 [8] An *infection chain* in the SIR model unfolds as follows. At most $k \in \mathbb{N}$ nodes may be infected by the Discoverer 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.

3 THE SOURCE DETECTION GAME

In our game, two players interact with each other. The *Discoverer* attempts to find a fixed source of infections and the *Adversary* controls the environment as well as the position of the source and attempts to conceal the source from the Discoverer. The goal of the Discoverer is to find the source while minimizing the number of successful infections until they find the source. We refer to this number as the *price of detection*. See Figure 1 for a formalization of the game.

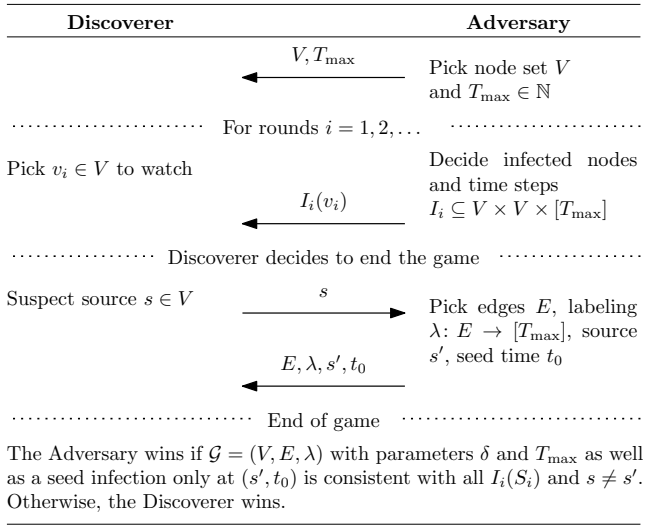

Figure 1: The SD game in the variation with consistent source behavior and known static graph.

Note that the formal model technically allows the adversary to change the location of the source at the end of the game, as long as the new location is consistent with the infection information retrieved by the Discoverer. We allow this in order to do a worst-case analysis, where the discoverer wins, if and only if, there is a single node possible where the source could be.

We investigate different restrictions placed upon the knowledge and power of the Discoverer and Adversary. The explored dimensions that impact the price of detection are:

- *Infection behavior*: Does the source have to infect in the same time step in each round, or can it infect in different time steps in each round? In any case, the Adversary must decide at which time step it is going to infect the source without knowing which node the Discoverer is watching.
- *Watched nodes*: How many nodes may the Discoverer watch in each round? We investigate the cases where the Discoverer watches one or two nodes. It then learns which of these nodes were infected, by which neighbor and when in the infection process started by the adversary.
- *Discoverer knowledge*: Which information about the graph is available to the Discoverer? We investigate the case where the Discoverer knows the underlying static graph and the

case where it only knows the nodes of the graph but not the edges.
- *Graph class*: How does the structure of the underlying static graph affects the price of detection? We look at general graphs, trees, and graphs of bounded treewidth.

Example 1. *Consider consistent source behavior (i.e., the source starts the infection at the same time in every round) and a known underlying graph. The graph is a path of length* 3*. Here, the Discoverer might choose to watch the middle node. Then, it can always detect the source after tolerating at most three infections. If, in the first round, at least two nodes get infected, the middle node must be infected and the Discoverer can deduce the source (by examining by which neighbor, if any, it was infected). If only one node gets infected in the first round, then, by consistency, the same must be true for the next two round. Thus, the Discover can spend rounds two and three watching the other two nodes to determine which one is the source while tolerating three infections in total.*

4 RANDOMIZED SOURCE DETECTION

Randomizing the behavior of the Discoverer enables us to evade the worst cases and yield good results with high probability. In fact, in this section, we even show that there is a randomized Discoverer that wins the SD game with consistent source behavior on unknown static graphs within $O(n\sqrt{n})$ infections and with constant success probability.

First, we give a useful primitive to use in the construction of more complex algorithms. We essentially prove that the source can, in expectation, not hide too many infections from the Discoverer. Looking at this fact the other way round, we see that if there is a linear number of infections, a Discoverer employing this strategy likely learns of at least one infection. Interestingly, this result holds for all settings.

Lemma 1. *Consider an instance of the SD game (either with known or unknown graph and either consistent or obliviously dynamic source behavior). Then there is a strategy for the Discoverer such that, after termination of the strategy, the Discoverer has observed a node in a round in which it gets infected. This strategy succeeds after tolerating at most* $3n$ *infections in expectation.*

Proof. In each round, the strategy picks a node to watch uniformly at random. It terminates when it has observed a node which gets infected while being watched. Let T be the random variable that takes the number of the round in which this happens. Let $a_1, \dots$ be the nodes infected in the respective rounds. Then the expected number of tolerated infections is $\mathbb{E}\left[\sum_{i=1}^{T} a_i\right] = \sum_{i=1}^{\infty} a_i \cdot \mathbb{P}\left[i \leq T\right] = \sum_{i=1}^{\infty} a_i \cdot \prod_{j=1}^{i-1} \left(\frac{n-a_j}{n}\right)$, where the equalities follow from the alternative definition of the expectation and since we require at least i rounds iff the first $i-1$, rounds are unsuccessful. Now, substitute $p_i = a_i/n$ for all i. Then p_i is the probability of finishing in round i assuming it is reached. We may also pull n outside the sum to view the number of tolerated infections in each round as a multiple of n. This leaves us with $n \sum_{i=1}^{\infty} p_i \cdot \prod_{j=1}^{i-1}(1-p_j)$. Now group the rounds of our strategy and thus the p_i by taking consecutive elements until their sum is at least $1/2$ and then start the next group. Formally, let $\{k_j\}_{j\in\mathbb{N}}$ be the (possibly finite) sequence such that k_j is the smallest integer such that $\sum_{i=k_j}^{k_{j+1}-1} p_i \geq 1/2$. Then, by union bound, the probability of finishing within a given group, assuming we reach it, is at least $1/2$. Also, we have that $\sum_{i=k_j}^{k_{j+1}-1} p_i \leq 3/2$ since all but the last elements together are less than $1/2$ and the last element is at most 1. This is a bound on the tolerated infections (as a multiple of n). Thus, we can bound the above expectation from above by the geometric process that tolerates $3n/2$ infections to perform the rounds in a given group and has probability $1/2$ of finishing within that block, and get

$$\leq n \sum_{j=1}^{\infty} \left(\sum_{i=k_j}^{i=k_{j+1}-1} p_i \right) \cdot \left(\prod_{\ell=1}^{j-1} \left(1 - \sum_{i=k_\ell}^{i=k_{\ell+1}-1} p_i \right) \right)$$

$$\leq n \sum_{j=1}^{\infty} 3/2 \cdot 1/2^{j-1} = 3n. \qquad \square$$

By Markov's inequality, the probability that this strategy takes at most $6n$ infections is at least constant ($1/2$ to be precise). Note that in the case of consistent source behavior, this yields a node sampled uniformly from the set of nodes which are infected as we employ uniform rejection sampling.

Algorithm 1: The randomized SD algorithm for unknown static graphs and consistent source behavior.

1. Find $\sqrt{n}$ nodes which are infected using repeated application of Lemma 1.
2. Now let a be the node among those picked that is infected the earliest.
3. Recursively, pick the next a as the node that infected the previous a.
4. If there is no neighbor left that gets infected earlier, we must have found the root.

Notice that this algorithm always finds the source and has a constant probability of terminating within $2\sqrt{n}$ rounds. Modifying step 3 such that it aborts after $\sqrt{n}$ rounds yields an algorithm that always terminates within $2\sqrt{n}$ rounds and has a constant probability of finding the source.

With this tool in hand, we can now give Algorithm 1.

⋆ Theorem 1. *Algorithm 1 solves the SD problem with consistent source behavior on unknown static graphs within* $O(n\sqrt{n})$ *infections with constant probability.*

Proof sketch. We first show that, picking a node close enough to the source in Step 1, results in finding the source in Steps 3 and 4 in $O(n\sqrt{n})$ rounds by examining the tree of the infection behavior. We bound the probability of picking a node close enough to the source by showing that not too many nodes can be far away from the source (since being far away requires a number of intermediate nodes). $\square$

Now that we have an upper bound on unknown general graphs, the pressing question is if it can be asymptotically improved. The answer is no and not even if the underlying graph is a tree. Note

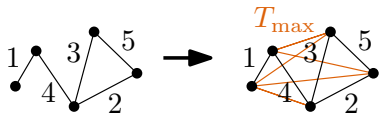

Figure 2: Construction from the proof of Theorem 3. Discoverer for known graphs A is used on the (right) completed graph to win the game in the setting of unknown graphs.

that since trees are the more restricted graph class, the following lower bound applies directly to general graphs, too.

★ Theorem 2. *Let $0 < \varepsilon < 1$ be arbitrarily small. For any algorithm solving the SD problem on trees with consistent source behavior, unknown static graph and watching a single node with success probability $p > 0$, there is an instance for every number of nodes n such that the price of detection under that algorithm is at least $(1/2) \cdot \sqrt{\ln\left((1 - p/(2+\varepsilon))^{-2}\right)} n\sqrt{n}$.*

This also gives us a bound on the trade-off between the success probability and the price of detection. For a more intuitive bound, obverse that by simple analytic tools (or a computer algebra system) we can deduce that even a price of detection of $pn\sqrt{n}$ is unachievable for any algorithm with success probability $p < 0.8724$.

Proof sketch. For $n \in \mathbb{N}^+$, we consider the path on n nodes labeled $\lambda(i, i+1) \coloneqq i$ for $i \in [n-1]$. We assume that there is an algorithm A that wins the game on these graphs within $c\sqrt{n}$ rounds with probability at least p. We construct algorithm B with strictly stronger capabilities: in each round, B watches the same nodes as A and also all nodes that A has observed to become infected in previous rounds. We argue that for A to win the game, B has to win the game, which happens if A guesses a node which is close to the source. We bound the probability of that event to obtain the result. □

5 WHEN KNOWING THE STATIC GRAPH HELPS

One would expect that knowing the static graph where the infections take place puts the Discoverer at a significant advantage. Surprisingly, we prove that generally, this is not the case. Motivated by this negative result, we investigate which knowledge about the static graph has value for the Discoverer. We show that if the static graph has treewidth tw, the price of detection is in $O(\mathrm{tw} \cdot n \log n)$. Finally, we argue that the central property that allows this is that the graph then recursively has small separators if it has small treewidth.

★ Theorem 3. *Let A be a Discoverer algorithm for the SD game with consistent source behavior on known static graphs. Then there is an algorithm A^u for the SD game with consistent source behavior on unknown static graphs such that if A wins the game within $O(f(n))$ infections with probability $g(n)$, then so does A^u for its game.*

The main idea of this reduction is to use the algorithm A as a subroutine by reporting $\binom{V}{2}$ as the edge set to it.

After this negative result, we explore when knowing the static graph does help. In particular, this also gives us a structural insight into which kinds of static graphs are easy to discover the time labels on.

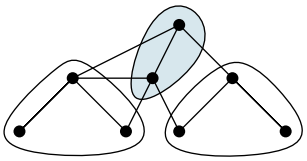

Figure 3: A graph with a balanced separator of size 2 (blue). An infection chain that includes nodes from both the left and right partitions must pass through the separator. Thus, in these cases, watching the separator reveals which of the partitions the source is in.

★ Theorem 4. *There is an algorithm that wins the SD game with consistent source behavior on known graphs while only tolerating $O(\mathrm{tw} \cdot n \log n)$ infections in expectation, where* tw *is the treewidth of the graph.*

This result is achieved by Algorithm 2. It crucially depends on the existence of small separators. Specifically, in a static graph G with treewidth tw, there is a $1/2$-balanced separator S (i.e., every connected component in $G - S$ has size at most $1/2\,|V(G)|$) of size at most tw + 1 [7]. See Figure 3 for an illustration. This result also holds if the nodes are weighted, and we ask for a separator where each component has at most half of the weight. Therefore, if we assign either 0 or 1 as weights, we pick which nodes we want to evenly distribute on the sides of the separator.

Algorithm 2: The SD algorithm for static graphs with bounded treewidth and consistent source behavior.

Maintain a set of nodes that could still be the source.
Until the watched node is the source, repeat:
1. Compute a balanced (tw + 1)-size separator of the candidate nodes.
2. Watch the separator nodes for one round each.
3. If none of them get infected, watch one of the candidate nodes uniformly at random until one is infected.
4. Update the candidate nodes to only include nodes on the component induced by removing the separator that first infected a node in the separator.

Proof sketch. The claimed properties of the algorithm follow, since these three properties hold after each iteration:

(1) We correctly track the candidate nodes (i.e., the source must always be one of the remaining candidates).
(2) After each iteration of the main loop of the algorithm, the number of nodes halves.
(3) With constant probability, we only tolerate a linear number of infections until detection. □

Intuitively, this dependence on the existence of separators makes sense. Looking for the source means that in each round, we have to decide where to look next based on the information gleamed so far. Together, Theorem 1 and Theorem 2 show that in the case of unknown graphs, we cannot do much better than testing a few nodes and then retracing their infection path node-by-node. In

some cases, we can do better by performing a binary search for the source, but in order to decide which half of our candidate nodes infections stem from, we have to separate them. Such separators may not be too large since we need to spend rounds on watching them, thus this technique is only useful if small separators exist recursively. This explains the connection between small treewidth and a smaller price of detection.

6 THE SPECIAL CASE OF TREES

First, observe that our algorithm exploiting the treewidth of a known graph (see Theorem 4) directly translates to trees.

★ Corollary 1. *There is an algorithm that wins the SD game with consistent source behavior on known trees while only tolerating $O(n \log n)$ infections with constant probability.*

For trees, we also show a strong matching lower bound.

Theorem 5. *There is an infinite family of trees $\{P_i\}_{i\in\mathbb{N}}$ such that, for any algorithm winning the SD game with consistent source behavior and known static graph on tree graphs with success probability $p > 0$, the price of detection under that algorithm is asymptotically equivalent to $n \log n$.*

This implies that there is no algorithm winning this game setup in $o(n \log n)$ with constant probability. Again, the proof can be adjusted to instead hold for algorithms that always win the game and have at least a constant probability to finish with less than $o(n \log n)$ infections.

Surprisingly, the concrete success probability does not play a role for the result (as long as it is positive and constant). Meaning that, for large enough graphs, we cannot trade a lower success probability for a multiplicative improvement over the $n \log n$ number of tolerated infections. Compare that to the weaker lower bound from Theorem 2, where this door is left open (though there is no known algorithm to exploit it).

Proof. First of all, let $n \in \mathbb{N}$. Then set P_n as the path with n nodes. Set $\delta = n$. Now let A be any algorithm that wins the SD game with consistent source behavior on known trees in $o(n \log n)$ rounds with a constant probability p. The adversary picks the source node s uniformly at random. Then, let the adversary set the edge labels for all $i < s$ as $\lambda(i, i+1) = n - i$ and for all $i \geq s$ as $\lambda(i, i+1) = i$.

Note that we require the algorithm to solve the problem on any tree and with any source with constant probability. We now model all possible randomized algorithms as a distribution over what we call execution trees. These execution trees encode the reactions the algorithm may have to all possible responses by the adversary. An instance of the game is then equivalent to picking an execution tree at the beginning and evaluating the responses of the adversary against the execution tree, yielding a path through the execution tree.

The behavior of the Discoverer can be described as the series of nodes it picks to watch. We may assume w.l.o.g. that the last node the Discoverer picks is the one it believes is the root (this is similar to the proof of Theorem 2). Now, by definition of the game, in each round, the Discoverer submits a node to watch and receives information about when and from which direction it was infected (if at all). By definition of our instances, every node is infected in every round. Thus, the only information the Discoverer receives is from which direction a node was infected and at which time step.

The edge labels only encode whether the edge is on the left or right side of the source, which the Discoverer also learns because it learns the direction along which the infection travels over the edge. Thus, we may disregard the effect that this knowledge has on the behavior of the Discoverer.

We construct this (binary) tree as follows. As the Discoverer algorithm is randomized, which node it picks in the next rounds, is dependent on previous information and random choices. We can thus assume the Discoverer makes all its random choices at the start of the execution and then builds a tree of possible outcomes. Each tree node is labeled with the node the Discoverer will watch in a certain round. Then, the left and right children are the choices, the Discoverer algorithm will make if the watched node is infected by its left and right neighbor respectively.

Thus, after picking an execution tree, the behavior of the Discoverer algorithm is deterministic. To complete the result, we show that for any possible execution tree with height in $o(\log n)$, there are a sufficient amount of source nodes for which the algorithm would not win. Clearly, an execution of the game is now associated with the execution tree the Discoverer algorithm picks at the start and the path through that tree. Since we assumed the algorithm always watches the source node in at least one round, an execution is only winning if the path taken includes the source nodes.

Assume the algorithm terminates after r rounds and let $c \in \mathbb{R}^+$ such that $r = c \log n$. We may assume that the execution path has length at most r. Since the Discoverer can now only win if the source is in the r first layers of the execution tree (in which there are 2^r nodes) and the source was picked uniformly at random, we have that $2^r/n \geq p$. Thus, $2^{c \log n}/n \geq p$, which rearranges to $n^c/n \geq p$. Applying the logarithm and rearranging leaves us with $c \geq 1 + \log p / \log n$.

The right-hand side approaches 1 as n approaches ∞. Thus, r approaches $\log n$ and since on our family of graphs, all nodes are infected in each round, the number of tolerated infections is asymptotically equivalent to $n \log n$. □

7 DYNAMIC INFECTION BEHAVIOR

In this variation, the Adversary may change the time where the source is seed-infected between rounds, but not the source itself. The Discoverer still learns when the watched node becomes infected each round. Note that the Adversary does not learn which nodes the Discoverer watches, and so its behavior may not depend on the choices of the Discoverer. We will see that for this variation, the simple deterministic brute-force algorithm is asymptotically optimal.

★ Theorem 6. *The brute-force Discoverer wins any instance of the SD game on unknown graphs and with obliviously dynamic source behavior that tolerates at most n^2 infections.*

This result is not randomized; the algorithm is always successful. As this variation is the most general, the result directly translates to all the other game variations. The result is indeed tight, as we see in the following theorem. We cannot even improve upon the quadratic cost of detection if we only ask for an algorithm with a constant probability of success.

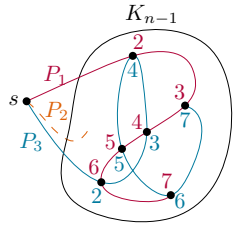

Figure 4: Construction from the proof of Theorem 7. The source is s and P_1, P_2, P_3 are Hamiltonian paths. For clarity, P_2 is omitted. Nodes are labeled based on P_1 and P_3.

Theorem 7. *For any algorithm solving the SD problem with obliviously dynamic behavior, known static graph, and watching a single node with success probability $p > 0$, there is an infinite family of graphs such that the price of detection under that algorithm is at least* $\min(1/3, p/2)n^2$.

Proof. Let $n \in 2\mathbb{N}+1$. We construct G as follows: take $(n-1)/2$ edge-disjoint Hamiltonian paths $P_1, \ldots, P_{(n-1)/2}$ on the K_{n-1} (which exist by [1]), add a node s and connect s to the first node on each of the Hamiltonian paths. From now on, we interpret each P_i as starting at s. Order the nodes and the Hamiltonian paths uniformly at random. Note, these cover all edges in $\mathcal{G}$. Write $p\colon E(\mathcal{G}) \to [n/2]$ be the function that maps an edge to the Hamiltonian path it belongs to and write $q\colon E(\mathcal{G}) \to [n]$ for the function that assigns an edge its index on that path. Then let the Adversary act according to $\lambda := e \mapsto p(e)n + q(e)$, and pick $\delta = 1$. See Figure 4 for an illustration of this construction. In each round i, the Adversary lets the source infect the whole graph via $P_{i \bmod (n-1)/2}$. For the sake of this proof, we may assume the Discoverer knows the scheme according to which these paths are picked (but not in which order the nodes appear on a specific path). Thus, if the Discoverer knows the label of an edge e, it may deduce which path it belongs to (by calculating $\lfloor \lambda(e)/n \rfloor$) and what the index of the edge has on the path (by calculating $\lambda(e) \bmod n$).

We only consider the first $(n-1)/2-1$ rounds, since after these, there will have been $n((n-1)/2-1) \in \Omega(n^2)$ infections. This is where the $1/3$ in the minimum in this theorem comes from, since for large enough n, we have $n((n-1)/2-1) > 1/3 \cdot n^2$. This case handled, we assume for the rest of this proof, that during our game there is exactly one infection chain per path.

Intuitively, the Discoverer has information about some of the edges for each path, and the goal of the Discoverer is to find the start of any of the paths. Formally, we prove that the Discoverer must have observed an infection involving the source (i.e., the first node on all paths), that is, it must have watched the first or second node on a path. By construction, an observed infection on one path reveals very limited information about the other paths: it reveals the positions of the two involved nodes on the current path. This only tells the Discoverer that they may not be adjacent on any other path. Within the first $(n-1)/2-1$ rounds, the Discoverer learns precisely $(n-1)/2-1$ of these pieces of information, thus for each node, there are at least $n/2$ positions on the current path that may be consistent with this information. Also, after $(n-1)/2-1$ rounds, there are at least two nodes on which the Discoverer has no information. Thus, if the Discoverer never observes the first or second node on a path, the adversary may rearrange the paths in such a way, that one of the unobserved nodes becomes the source (it picks whichever node the Discoverer does not claim is the source).

Since the positions are distributed uniformly at random, no strategy that the Discoverer uses has a probability larger than $2/n$ of picking the first or second node on the current path, thus advancing the requirements outlined above. Any algorithm that has success probability at least p, must have at least probability p of picking the first or second node on a path. By union bound, the Discoverer must have spent at least $np/2$ rounds, that is, tolerated $n^2p/2$ infections. □

★ Theorem 8. *There is a Discoverer algorithm that wins any instance of the SD game on known tree graphs under obliviously dynamic source behavior while watching at most two nodes per round. This algorithm tolerates $O(n \log n)$ infections in expectation.*

Proof sketch. Similar to Algorithm 2, we introduce an algorithm that relies on the existence of small spanners. Concretely, it is a well known result that a tree has a *centroid decomposition* [14]. A centroid is a node such that its removal disconnects the tree into two components at most half the size of the original tree. Since, any tree has a centroid and the two components are again trees, we can decompose a tree into a series of these centroids and the respective split trees. The algorithm now proceeds as follows:

(1) Maintain a subtree of candidate nodes, and always compute a centroid as a balanced separator. Start with the whole graph as the candidates.
(2) In each round: watch the current separator and one node in the current subtree picked uniformly at random.
   (a) If the separator is infected, recurse into the side of the separation from which the infection originated.
   (b) If the random node is infected but not the separator, recurse into the separation side the node is a part of. □

Note that this is the only result in this paper that depends on the Discoverer's ability to watch more than one node at a time. As we will see in the next theorem, this can only be necessary if the source has obliviously dynamic behavior. We currently do not know of a lower bound proving that watching more than one node in these scenarios provides an asymptotic advantage. Intuitively, the above algorithm exploits this capability to mitigate one of the problems with dynamic source behavior: we cannot trace back along an infection chain, as it is unclear if in a given round we see similar behavior to the previous one. By watching both the next and previous node to be tested, we can extenuate this issue.

★ Theorem 9. *Let A be a Discoverer algorithm for the SD game with consistent source behavior (with either known or unknown static graph) while watching k nodes per round. Then there is a Discoverer algorithm A^1 such that if A tolerates at most x successful infections in some instance of the game, A^1 tolerates at most kx infections.*

This proves that, asymptotically, the ability to watch two nodes does not help the Discoverer in the setting with consistent source behavior. Note, this theorem and its proof can trivially be extended to randomized bounds.

8 OUTLOOK

Our work could naturally be extended by studying SD under the susceptible-infected-susceptible model. As many of our lower bounds rely on the resistance of recovered nodes, seeing which results translate promises to be insightful. Closing the remaining gap in Table 1 would require investigating SD on unknown graphs with obliviously dynamic source behavior where the underlying static graph is guaranteed to be a tree.

Finally, the most compelling unanswered question is whether allowing the Discoverer to watch multiple nodes in the setting with obliviously dynamic source behavior allows for asymptotically fast algorithms. We know this is not the case in the setting with consistent source behavior by Theorem 9. Yet, while we use the additional capability in Theorem 8, we have no proof it is fundamentally necessary to achieve the better running time.

REFERENCES

[1] Kyriakos Axiotis and Dimitris Fotakis. 2016. On the Size and the Approximability of Minimum Temporally Connected Subgraphs. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, Vol. 55. 149:1–149:14. https://doi.org/10.4230/LIPIcs.ICALP.2016.149

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

[3] Tom Britton. 2010. Stochastic epidemic models: A survey. *Mathematical Biosciences* 225, 1 (2010), 24–35. https://doi.org/10.1016/j.mbs.2010.01.006

[4] Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. 2011. Limiting the spread of misinformation in social networks. *Proceedings of the 20th international conference on World wide web* (2011), 665–674.

[5] Yi-Cheng Chen, Ping-En Lu, Cheng-Shang Chang, and Tzu-Hsuan Liu. 2020. A Time-Dependent SIR Model for COVID-19 With Undetectable Infected Persons. *IEEE Transactions on Network Science and Engineering* 7, 4 (2020), 3279–3294. https://doi.org/10.1109/TNSE.2020.3024723

[6] Ian Cooper, Argha Mondal, and Chris G. Antonopoulos. 2020. A SIR model assumption for the spread of COVID-19 in different communities. *Chaos, Solitons & Fractals* 139 (2020), 110057. https://doi.org/10.1016/j.chaos.2020.110057

[7] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. 2015. *Parameterized algorithms*. Vol. 5. Springer, 193.

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

[9] Stephanie Forrest and Catherine Beauchemin. 2007. Computer immunology. *Immunological Reviews* 216, 1 (2007), 176–197. https://doi.org/10.1111/j.1600-065X.2007.00499.x

[10] Maksym Gabielkov, Arthi Ramachandran, Augustin Chaintreau, and Arnaud Legout. 2016. Social Clicks: What and Who Gets Read on Twitter?. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science* (Antibes Juan-les-Pins, France) *(SIGMETRICS '16)*. Association for Computing Machinery, New York, NY, USA, 179–192. https://doi.org/10.1145/2896377.2901462

[11] Nathalie T.H. Gayraud, Evaggelia Pitoura, and Panayiotis Tsaparas. 2015. Diffusion Maximization in Evolving Social Networks. In *Proceedings of the 2015 ACM on Conference on Online Social Networks* (Palo Alto, California, USA) *(COSN '15)*. Association for Computing Machinery, New York, NY, USA, 125–135. https://doi.org/10.1145/2817946.2817965

[12] Herbert W. Hethcote. 1989. *Three Basic Epidemiological Models*. Springer Berlin Heidelberg, Berlin, Heidelberg, 119–144. https://doi.org/10.1007/978-3-642-61317-3_5

[13] Qiangjuan Huang. 2017. Source Locating of Spreading Dynamics in Temporal Networks. In *Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, April 3-7, 2017*, Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich (Eds.). ACM, 723–727. https://doi.org/10.1145/3041021.3053376

[14] Camille Jordan. 1869. Sur les assemblages de lignes. *Journal für die reine und angewandte Mathematik* 1869, 70 (1869), 185–190. https://doi.org/doi:10.1515/crll.1869.70.185

[15] David Kempe, Jon Kleinberg, and Amit Kumar. 2000. Connectivity and inference problems for temporal networks. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*. 504–513.

[16] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2003), 137–146. https://doi.org/10.1145/956750.956769

[17] William Ogilvy Kermack, A. G. McKendrick, and Gilbert Thomas Walker. 1927. 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 (1927), 700–721. https://doi.org/10.1098/rspa.1927.0118

[18] Nikolay A. Kudryashov, Mikhail A. Chmykhov, and Michael Vigdorowitsch. 2021. Analytical features of the SIR model and their applications to COVID-19. *Applied Mathematical Modelling* 90 (2021), 466–473. https://doi.org/10.1016/j.apm.2020.08.057

[19] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Jose, California, USA) *(KDD '07)*. Association for Computing Machinery, New York, NY, USA, 420–429. https://doi.org/10.1145/1281192.1281239

[20] Hung T. Nguyen, My T. Thai, and Thang N. Dinh. 2016. Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-scale Networks. In *Proceedings of the 2016 International Conference on Management of Data* (San Francisco, California, USA) *(SIGMOD '16)*. Association for Computing Machinery, New York, NY, USA, 695–710. https://doi.org/10.1145/2882903.2915207

[21] Robert Paluch, Xiaoyan Lu, Krzysztof Suchecki, Bolesław K. Szymański, and Janusz A. Hołyst. 2018. Fast and accurate detection of spread source in large complex networks. *Scientific Reports* 8, 1 (Feb. 2018), 2508. https://doi.org/10.1038/s41598-018-20546-3

[22] Pedro C Pinto, Patrick Thiran, and Martin Vetterli. 2012. Locating the source of diffusion in large-scale networks. *Physical review letters* 109, 6 (2012), 068702.

[23] Polina Rozenshtein, Aristides Gionis, B. Aditya Prakash, and Jilles Vreeken. 2016. Reconstructing an Epidemic Over Time. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (Eds.). ACM, 1835–1844. https://doi.org/10.1145/2939672.2939865

[24] Devavrat Shah and Tauhid Zaman. 2011. Rumors in a network: Who's the culprit? *IEEE Transactions on information theory* 57, 8 (2011), 5163–5181.

[25] Karishma Sharma, Feng Qian, He Jiang, Natali Ruchansky, Ming Zhang, and Yan Liu. 2019. Combating Fake News: A Survey on Identification and Mitigation Techniques. *ACM Trans. Intell. Syst. Technol.* 10, 3, Article 21 (apr 2019), 42 pages. https://doi.org/10.1145/3305260

[26] Kai Zhu and Lei Ying. 2016. Information Source Detection in the SIR Model: A Sample-Path-Based Approach. *IEEE/ACM Transactions on Networking* 24, 1 (2016), 408–421. https://doi.org/10.1109/TNET.2014.2364972