Linear-Time Exact Computation of Influence Spread on Bounded-Pathwidth Graphs

Kengo Nakamura
Communication Science Laboratories, NTT, Inc., Kyoto, Japan

Masaaki Nishino
Communication Science Laboratories, NTT, Inc., Kyoto, Japan

Abstract

Given a network and a set of vertices called seeds to initially inject information, influence spread is the expected number of vertices that eventually receive the information under a certain stochastic model of information propagation. Under the commonly used independent cascade model, influence spread is equivalent to the expected number of vertices reachable from the seeds on a directed uncertain graph, and the exact evaluation of influence spread offers many applications, e.g., influence maximization. Although its evaluation is a #P-hard task, there is an algorithm that can precisely compute the influence spread in $O(mn\omega_p^2 \cdot 2^{\omega_p^2})$ time, where ω_p is the pathwidth of the graph. We improve this by developing an algorithm that computes the influence spread in $O((m+n)\omega_p^2 \cdot 2^{\omega_p^2})$ time. This is achieved by identifying the similarities in the repetitive computations in the existing algorithm and sharing them to reduce computation. Although similar refinements have been considered for the probability computation on undirected uncertain graphs, a greater number of similarities must be leveraged for directed graphs to achieve linear time complexity.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Networks → Network algorithms

Keywords and phrases Influence spread, bounded pathwidth, network reliability, linear time algorithm

Digital Object Identifier 10.4230/LIPIcs.SWAT.2026.34

Related Version *Full Version*: https://arxiv.org/abs/2604.13526

Funding This work was supported by JSPS KAKENHI Grant Number JP26K02906.

1 Introduction

Given a network and *seed set* S, which is a set of vertices that are initially injected information, *influence spread* [5] $\sigma(S)$ is the expected number of vertices that eventually receives the information under a stochastic model of information propagation. Influence spread is implemented in viral marketing applications with the aim of evaluating the influence of a particular node within a social network. Currently, it is also a keystone for other network applications such as network monitoring [11], rumor control [23], target advertisement [13], and social recommendation [24]. Among several information diffusion models, the most basic and commonly used one is the *independent cascade (IC) model* [6]. In this model, a network is modeled as a directed graph $G = (V, E)$, where each edge $e \in E$ is associated with a probability p_e. When a vertex v receives information, it stochastically propagates the information along every outgoing edge e independently with a given probability p_e.

The influence spread $\sigma(S)$ under the IC model is equivalent to the expected number of vertices reachable from at least one vertex in S on a directed *uncertain graph*. Given directed graph $G = (V, E)$ and probability p_e for every edge $e \in E$, we consider an uncertainty in which each edge $e \in E$ is present with probability p_e and absent with probability $1 - p_e$. We assume that each edge's presence or absence is stochastically independent of the other edges.

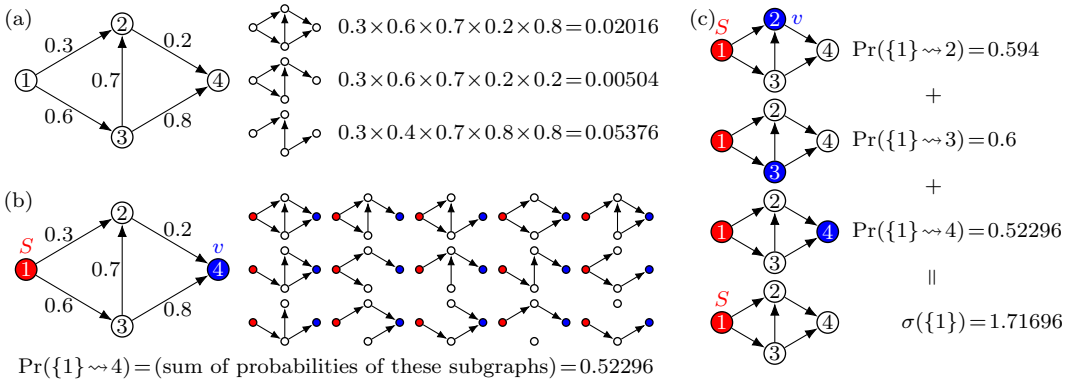

Figure 1 (a) Example of directed uncertain graph and probabilities of subgraphs. (b) All 15 subgraphs in which $v = 4$ is reachable from $\{S\} = 1$. $\Pr(\{1\} \rightsquigarrow 4)$ is the sum of probabilities of these subgraphs. (c) $\sigma(\{1\})$ is computed as the sum of $\Pr(\{1\} \rightsquigarrow v)$s for $v = 2, 3, 4$.

Accordingly, the probability that a subgraph (V, E') $(E' \subseteq E)$ appears is

$$\Pr(E') \coloneqq \prod_{e \in E'} p_e \cdot \prod_{e \in E \setminus E'} (1 - p_e). \tag{1}$$

Here, a directed path from some $s \in S$ to vertex v on the subgraph (V, E') corresponds to the route of information propagation from s to v. Therefore, given $S \subseteq V$ and $v \in V$, the probability that v receives information is equal to the probability $\Pr(S \rightsquigarrow v)$ that v can be reached from some vertices in S under uncertainty. Here, $\Pr(S \rightsquigarrow v) = \sum_{E' \in \mathcal{E}_{S \rightsquigarrow v}} \Pr(E')$, where $\mathcal{E}_{S \rightsquigarrow v}$ is the family of subsets E' of edges such that v can be reached from some vertices in S on (G, E'). Finally, the influence spread can be represented as $\sigma(S) = \sum_{v \in V \setminus S} \Pr(S \rightsquigarrow v)$.

▶ **Example 1.1.** Figure 1a depicts examples of the probabilities $\Pr(E')$ of subgraphs (V, E') $(E' \subseteq E)$. For this uncertain graph, there are 15 subgraphs in which $v = 4$ is reachable from $\{S\} = 1$ (Figure 1b); thus, $\Pr(\{1\} \rightsquigarrow 4)$ is the sum of $\Pr(E')$ over all these subgraphs. By evaluating $\Pr(\{1\} \rightsquigarrow 2)$, $\Pr(\{1\} \rightsquigarrow 3)$, and $\Pr(\{1\} \rightsquigarrow 4)$ as in Figure 1c, $\sigma(\{1\})$ can be computed as their sum.

Since evaluating the influence spread $\sigma(S)$ is a #P-hard task [3], most studies have relied on Monte Carlo simulation for evaluating $\sigma(S)$ [7, 10, 11, 18]. However, exact computation of $\sigma(S)$ is important for the following reasons. First, this allows us to more accurately evaluate the influence spread in larger networks, since real networks often consist of small communities; the exact computation of influence spread in each small community will improve overall evaluation quality. Second, this allows us to rank the influential vertices in descending order of influence spread. For this purpose, Monte Carlo simulations are not sufficient because $\Omega(1/\varepsilon^2)$ samples are needed to obtain $(1 \pm \varepsilon)$-approximation with high probability [18]; to distinguish every vertex's influence spread, high accuracy such as $\varepsilon \sim 10^{-5}$ is sometimes needed, which is costly for Monte Carlo methods.

Maehara et al. [15] proposed the only non-trivial algorithm to exactly compute $\sigma(S)$ under the IC model. Given seed set S and v, it efficiently computes $\Pr(S \rightsquigarrow v)$, i.e., the probability that v receives information. Given the path decomposition of G whose width is ω_p, this probability for a single v can be computed in $O(m\omega_p^2 \cdot 2^{\omega_p^2})$ time, where m is the number of edges. Hereafter, we mean the pathwidth of a directed graph G in the sense of the pathwidth of the underlying undirected graph of G as used in the literature [15, 20]. The influence spread $\sigma(S) = \sum_{v \in V \setminus S} \Pr(S \rightsquigarrow v)$ can be computed in $O(mn\omega_p^2 \cdot 2^{\omega_p^2})$ time, where n is the number of vertices.

Here, even for bounded-pathwidth graphs, obtaining the $\sigma(S)$ value requires super-linear ($O(mn)$) time. Since there are similarities in the repetitive computation for different vertices v using this algorithm, we investigated the use of these similarities to refine their algorithm and thus improve the running time for computing $\sigma(S)$.

1.1 Our Contribution

In this paper, we propose an algorithm for computing $\sigma(S)$ for given S under the IC model. Specifically, the proposed algorithm simultaneously computes $\Pr(S \rightsquigarrow v)$ in an uncertain directed graph for all vertices v. Our main result can be expressed as follows.

▶ **Theorem 1.2.** *Given directed graph G, seed set S, each edge's probability p_e of presence, and the path decomposition of G whose width is ω_p, the probability $\Pr(S \rightsquigarrow v)$ for every vertex v can be computed in $O((m+n)\omega_p^2 \cdot 2^{\omega_p^2})$ time in total.*

Since even inputting the entire graph requires linear time, this algorithm is asymptotically optimal for graphs with a bounded pathwidth when the constant factor is ignored. Moreover, the exponential factor of the pathwidth, hidden in the constant, remains the same as in the algorithm of Maehara et al. [15]: the dependence on the graph sizes is improved from $O(mn)$ to $O(m+n)$ while the dependence on the pathwidth is kept the same.

Our algorithm shares certain ideas with the existing algorithm for simultaneously computing the probabilities of connection in undirected graphs [16] in that both use similarities in the computation of the previous algorithms [8, 15] for different vertices. However, there is a non-trivial gap in extending that algorithm [16] to directed graphs due to the difference between connectivity in undirected graphs and reachability in directed graphs. We fill this gap by finding additional similarities in computing $\sigma(S)$ with the previous algorithm [15]; details are discussed in Section 5.

1.2 Related Work

Influence maximization. The most well-studied problem related to influence spread is influence maximization [10], that is, the problem of choosing a seed set S to maximize the influence spread. This has attracted much attention over the past two decades [12, 14, 19]. Although the influence maximization problem under most diffusion models including the IC model is NP-hard [10], a simple greedy algorithm [10] can achieve a $(1-1/e)$-approximation provided that every $\sigma(S)$ is computed exactly. Various methods, e.g., the sketch-based methods described in a survey [12], have improved this greedy algorithm. However, if we resorted to using Monte Carlo simulation to evaluate $\sigma(S)$, we could not obtain a deterministic approximation bound for influence maximization, even with greedy or more sophisticated algorithms. Moreover, although many studies have used an alternative proxy for $\sigma(S)$ to speed up the computation (e.g., proxy-based methods [12] and learning-based methods [14]), they also lack approximation bounds. This situation accentuates the theoretical importance of an exact evaluation of influence spread.

Network reliability. Computing the probability that some vertices are connected in an uncertain graph has been traditionally studied as *network reliability* evaluation in the network community. Most studies on network reliability evaluation have modeled the network as an *undirected* graph and, given a vertex set K and the probability p_e of presence for every edge, they computed the probability that the vertices in K are connected. Even for undirected graphs, computing this probability is #P-complete [22]. Nevertheless, Hardy et al. [8] developed an algorithm that could compute this probability

in linear time for graphs with bounded pathwidth. Another line of research has attempted to compute the expected number of vertices connected to vertex u [17], which is an analogue of the influence spread under the IC model for undirected graphs. Nakamura et al. [16] proposed an algorithm that could compute the probability that a vertex u is connected to v for *every* $v \in V$ in linear time for an undirected graph with bounded pathwidth. By summing all the probabilities for every v, it is also possible to compute the expected number of vertices connected to v in linear time. This algorithm uses the similarities in the computation of the probability that u and v are connected with Hardy's algorithm [8] for different v; therefore, it shares similar ideas with our work. However, we cannot straightforwardly extend this algorithm to handle directed graphs, as described in Section 1; detailed discussions are given in Section 5. Note that these algorithms [8, 16] also run in polynomial time for graphs with bounded pathwidth, indicating the importance of algorithms dealing with bounded-pathwidth graphs.

2 Preliminaries

For sets U, W of vertices, $U \rightsquigarrow_G W$ means that u can reach w, i.e., w can be reached from u, for some $u \in U$ and $w \in W$ in directed graph G. If U (or W) consists of only a single vertex u (w), we simply write, e.g., $u \rightsquigarrow_G W$ and $u \rightsquigarrow_G w$ instead of $\{u\} \rightsquigarrow_G W$ and $\{u\} \rightsquigarrow_G \{w\}$. When it is clear from the context, we omit the subscript G.

A directed graph is called *strongly connected* if $u \rightsquigarrow v$ for every ordered pair (u, v) of vertices. We say vertex subset C is a *strongly connected component* (SCC) of G if the vertex-induced subgraph of G induced by C is strongly connected but that induced by $V \cup \{v\}$ is not strongly connected for any $v \in V \setminus C$. It is known that, given directed graph G, all SCCs in G can be computed in linear time in the size of G [21]. Throughout this paper, n and m denote the number of vertices and edges, respectively, in the input graph G.

We formally define the problems to solve as follows.

▶ **Problem 2.1.** Given a directed graph G, probabilities $\{p_e\}_{e \in E}$, and a seed set $S \subseteq V$, compute the probability $\Pr(S \rightsquigarrow v)$ for every $v \in V \setminus S$.

For convenience, we assume that G contains no self-loops and the underlying undirected graph of G is connected. Note that if G contains self-loops, we can safely remove it, since it would never affect reachability. If G is disconnected, we can solve the problem individually for each connected component.

3 Review of Previous Algorithm

3.1 Decomposition of Probability

In the following, we review Maehara's algorithm [15] for exactly computing $\Pr(S \rightsquigarrow v)$ for a given S, v. Although the literature [15] only showed the algorithm for the case $|S| = 1$, here we show the general S case because it can be easily extended.

The basic idea in evaluating $\Pr(S \rightsquigarrow v)$ is to decompose $\Pr(S \rightsquigarrow v)$ into terms that can be easily computed. For $E', E'' \subseteq E$ such that $E' \cap E'' = \emptyset$, let $\Pr(\cdot \mid E', \overline{E''})$ be the conditional probability given that the edges in E' are present and those in E'' are absent. From the case analysis of whether $e \in E \setminus (E' \cup E'')$ is present or absent,

$$\Pr(S \rightsquigarrow v \mid E', \overline{E''}) = p_e \cdot \Pr(S \rightsquigarrow v \mid E' \cup \{e\}, \overline{E''}) + (1 - p_e) \cdot \Pr(S \rightsquigarrow v \mid E', \overline{E'' \cup \{e\}}). \quad (2)$$

By ordering edges as $e_1, \dots, e_m$, $\Pr(S \rightsquigarrow v)$ can be decomposed by recursively applying (2):

$$\begin{aligned}\Pr(S \rightsquigarrow v) &= p_{e_1} \cdot \Pr(S \rightsquigarrow v \mid \{e_1\}, \overline{\emptyset}) + (1-p_{e_1}) \cdot \Pr(S \rightsquigarrow v \mid \emptyset, \overline{\{e_1\}}) \\ &= p_{e_1}p_{e_2} \cdot \Pr(S \rightsquigarrow v \mid \{e_1, e_2\}, \overline{\emptyset}) + p_{e_1}(1-p_{e_2}) \cdot \Pr(S \rightsquigarrow v \mid \{e_1\}, \overline{\{e_2\}}) + \\ &\quad (1-p_{e_1})p_{e_2} \cdot \Pr(S \rightsquigarrow v \mid \{e_2\}, \overline{\{e_1\}}) + (1-p_{e_1})(1-p_{e_2}) \cdot \Pr(S \rightsquigarrow v \mid \emptyset, \overline{\{e_1, e_2\}}) \\ &= \cdots = \sum_{E' \subseteq E_{<i}} \Pr_{<i}(E') \cdot \Pr(S \rightsquigarrow v \mid E', \overline{E_{<i} \setminus E'}) = \cdots, \end{aligned} \tag{3}$$

$$\text{where} \quad E_{<i} \coloneqq \{e_1, \dots, e_{i-1}\}, \quad \Pr_{<i}(E') \coloneqq \prod_{e \in E'} p_e \cdot \prod_{e \in E_{<i} \setminus E'} (1 - p_e).$$

This expansion eventually reaches the definition $\Pr(S \rightsquigarrow v) \coloneqq \sum_{E' \in \mathcal{E}_{S \rightsquigarrow v}} \Pr(E')$. Therefore, by using this decomposition naively, we have $O(2^m)$ terms, which incurs exponential complexity of the graph size in evaluating it.

To prevent this, we attempt to detect the equality among probabilities, i.e., to find subsets $E', F' \subseteq E_{<i}$ $(E' \neq F')$ such that $\Pr(S \rightsquigarrow v \mid E', \overline{E_{<i} \setminus E'}) = \Pr(S \rightsquigarrow v \mid F', \overline{E_{<i} \setminus F'})$. If such equality can be detected, we simply need to further decompose only one among the equal probabilities, which reduces the number of terms. If the following condition (#) holds, we can confirm $\Pr(S \rightsquigarrow v \mid E', \overline{E_{<i} \setminus E'}) = \Pr(S \rightsquigarrow v \mid F', \overline{E_{<i} \setminus F'})$:

(#) For any $H \subseteq E_{\geq i} \coloneqq E \setminus E_{<i}$, $S \rightsquigarrow_{(V, E' \cup H)} v$ if and only if $S \rightsquigarrow_{(V, F' \cup H)} v$.

To check condition (#), we focus on the reachability relation on the subgraph (V, E'). If $u \rightsquigarrow_{(V,E')} w$ and $w \rightsquigarrow_{(V,F')} v$ are equivalent for any $u, w \in V$, we can confirm that (#) holds. In brief, this is because, given a path from $s \in S$ to v on $(V, E' \cup H)$, we can construct a path from s to v on $(V, F' \cup H)$ by using the above equivalence, and vice versa.

Maehara et al. [15] indicated that only a limited subset of vertices called *frontier vertices* is important for checking (#).

▶ **Definition 3.1.** *Given edge ordering $e_1, \dots, e_m$, (i-th)* frontier vertices W_i *are those appearing in both $E_{<i}$ and $E_{\geq i}$.*

We consider the reachability relations among $W_i \cup S \cup \{v\}$ on (V, E') as follows.

▶ **Definition 3.2.** *Let $W_i^{S \rightsquigarrow}(E') \coloneqq \{w \in W_i \mid S \rightsquigarrow_{(V,E')} w\}$ (frontier vertices reachable from S) and $W_i^{\rightsquigarrow v}(E') \coloneqq \{w \in W_i \mid w \rightsquigarrow_{(V,E')} v\}$ (those that can reach v). A binary matrix $\Phi_i^v(E')$ is defined as follows: (I) It is indexed by $(W_i \setminus W_i^{S \rightsquigarrow}(E') \cup \{S\}) \times (W_i \setminus W_i^{\rightsquigarrow v}(E') \cup \{v\})$. (II) $\Phi_i^v(E')_{uw} = 1$ if and only if $u \rightsquigarrow_{(V,E')} w$. Here, S is treated as a special (single) row index, and v is treated as a column index. We call $\Phi_i^v(E')$ a* transversal configuration *(TC).*

We can then prove the following.

▶ **Lemma 3.3.** *$\Phi_i^v(E') = \Phi_i^v(F')$ is a sufficient condition for (#), meaning that $\Pr(S \rightsquigarrow v \mid E', \overline{E_{<i} \setminus E'}) = \Pr(S \rightsquigarrow v \mid F', \overline{E_{<i} \setminus F'})$. Note that $\Phi_i^v(E') = \Phi_i^v(F')$ means that the indices of matrices as well as their entries are equal.*

This can be shown by transforming a u-v directed path on $(V, E' \cup H)$ for some $u \in S$ to a u'-v directed path on $(V, F' \cup H)$ for some $u' \in S$. Now we can rewrite $\Pr(S \rightsquigarrow v \mid E', \overline{E_{<i} \setminus E'})$ as $\Pr(S \rightsquigarrow v \mid \Phi_i^v(E'))$.

▶ **Example 3.4.** On the graph in Figure 2a with $v = 5$, we examine $\{e_1, e_4\}, \{e_1, e_2, e_3, e_4\}$, $\{e_1, e_2, e_4, e_6\}, \{e_2, e_5, e_6\} \subseteq E_{<8}$ corresponding to the four edge subgraphs of Figure 2b. These subsets have an identical TC Φ_8^5 because $W_8 = \{4, 5\}$. Thus, the conditional probabilities of $S \rightsquigarrow 5$ are all equal for these subsets.

Algorithm 1 Previous algorithm [15].

```
1  N'_1 ← {Φ^v_1(∅)}, N'_i ← {} (i ≥ 2)
2  for i ← 1 to m do                          // Diagram construction
3  |  foreach Φ ∈ N'_i do
4  |  |  foreach f ∈ {lo, hi} do
5  |  |  |  if f^v(Φ) = ⊥ then Φ.f ← ⊥
6  |  |  |  else if f^v(Φ) = ⊤ then Φ.f ← ⊤
7  |  |  |  else N'_{i+1} ← N'_{i+1} ∪ {f^v(Φ)},
   |  |  |    Φ.f ← f^v(Φ) ∈ N'_{i+1}
8  p[Φ] ← 1 for Φ ∈ N'_1, p[Φ] = 0 for Φ ∈ N'_i (i ≥ 2)
9  for i ← 1 to m do                          // Top-down DP
10 |  foreach Φ ∈ N'_i do                    // p[Φ] stores Pr(Φ)
11 |  |  p[Φ.lo] += (1 − p_{e_i}) · p[Φ]
12 |  |  p[Φ.hi] += p_{e_i} · p[Φ]
13 return p[⊤]                               // Pr(S ⇝ v) = Pr(⊤)
```

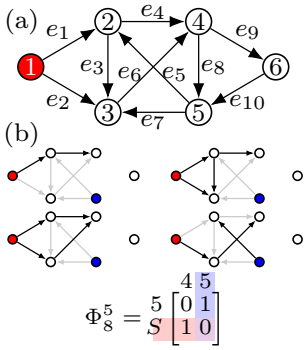

Figure 2 (a) Example of input graph G. Red vertex is included in S. (b) Example of different edge subsets of $E_{<8}$ leading to the same TC Φ_8^5.

Moreover, $\Phi_i^v(\cdot)$ has the following property: if $\Phi_i^v(E') = \Phi_i^v(F')$ for $E', F' \subseteq E_{<i}$, both $\Phi_{i+1}^v(E') = \Phi_{i+1}^v(F')$ and $\Phi_{i+1}^v(E' \cup \{e_i\}) = \Phi_{i+1}^v(F' \cup \{e_i\})$ hold. Thus, there exist transition functions $\mathsf{lo}^v, \mathsf{hi}^v$ from Φ_i^v to Φ_{i+1}^v satisfying $\mathsf{lo}^v(\Phi_i^v(E')) = \Phi_{i+1}^v(E')$ and $\mathsf{hi}^v(\Phi_i^v(E')) = \Phi_{i+1}^v(E' \cup \{e_i\})$ for any $E' \subseteq E_{<i}$. By combining this fact and (2), we have, for any $\Phi = \Phi_i^v(E')$ $(E' \subseteq E_{<i})$,

$$\Pr(S \rightsquigarrow v \mid \Phi) = p_e \cdot \Pr(S \rightsquigarrow v \mid \mathsf{hi}^v(\Phi)) + (1 - p_e) \cdot \Pr(S \rightsquigarrow v \mid \mathsf{lo}^v(\Phi)). \tag{4}$$

Using this decomposition recursively, $\Pr(S \rightsquigarrow v)$ can be decomposed as follows:

$$\begin{aligned} \Pr(S \rightsquigarrow v) &= \Pr(S \rightsquigarrow v \mid \Phi_1^v(\emptyset)) \\ &= p_{e_1} \cdot \Pr(S \rightsquigarrow v \mid \mathsf{hi}^v(\Phi_1^v(\emptyset))) + (1 - p_{e_1}) \cdot \Pr(S \rightsquigarrow v \mid \mathsf{lo}^v(\Phi_1^v(\emptyset))) \\ &= \cdots = \sum_{\Phi \in \mathcal{N}_i} \Pr(\Phi) \cdot \Pr(S \rightsquigarrow v \mid \Phi) = \cdots, \end{aligned} \tag{5}$$

$$\text{where} \quad \Pr(\Phi) \coloneqq \sum_{E' \subseteq E_{<i} : \Phi_i^v(E') = \Phi} \Pr_{<i}(E'), \quad \mathcal{N}_i \coloneqq \{\Phi_i^v(E') \mid E' \subseteq E_{<i}\}. \tag{6}$$

In other words, $\Pr(\Phi)$ is the probability that TC Φ appears and $\mathcal{N}_i$ is the set of all TCs of the subsets of $E_{<i}$. We later show that $|\mathcal{N}_i|$ is bounded by a constant if the pathwidth is bounded in Lemma 4.15. This is the main reason to achieve $O(m)$ time complexity in computing $\Pr(S \rightsquigarrow v)$.

For some TCs Φ, we can confirm that $S \rightsquigarrow v$ must hold or must not hold, i.e., $\Pr(S \rightsquigarrow v \mid \Phi) = 1$ or 0. For such TCs, we do not need to further decompose $\Pr(S \rightsquigarrow v \mid \Phi)$ with (4). We define such base cases with the following $\top$-pruning and $\bot$-pruning as follows. Let i_u be the largest index i such that one of the endpoints of e_i is u and let $i_S \coloneqq \max_{u \in S} i_u$.

- If $\mathsf{hi}^v(\Phi)_{Sv} = 1$, determining the presence of e_i confirms $S \rightsquigarrow v$, i.e., $\Pr(S \rightsquigarrow v \mid \mathsf{hi}^v(\Phi)) = 1$. In such a case, we let $\mathsf{hi}^v(\Phi) = \top$, a special TC meaning that $S \rightsquigarrow v$ must hold ($\top$-pruning).
- Suppose that $i \geq i_S$ and $\mathsf{lo}^v(\Phi)_{Su} = 0$ for any u, or that $i \geq i_v$ and $\mathsf{lo}^v(\Phi)_{uv} = 0$ for any u. In these cases, no further vertices can be reached from S or reach v by determining the absence of e_i, i.e., $\Pr(S \rightsquigarrow v \mid \mathsf{lo}^v(\Phi)) = 0$. Thus, we let $\mathsf{lo}^v(\Phi) = \bot$, which is a special TC meaning that $S \rightsquigarrow v$ must not hold ($\bot$-pruning)[1]. We also let $\mathsf{hi}^v(\Phi) = \bot$ if $\top$-pruning does not occur and $\mathsf{hi}^v(\Phi)$ satisfies the same condition as above.

[1] Although the original version [15] uses different pruning rules, they require a pre-computation of transitive closures of $(V, E_{\geq i})$, which takes cubic time. We use an alternative pruning rule that can be checked in linear time.

Eventually, all the TCs are transformed into either $\top$ or $\bot$, and (5) reaches $\Pr(S \rightsquigarrow v) = \Pr(\top) \cdot 1 + \Pr(\bot) \cdot 0 = \Pr(\top)$. Thus, computing $\Pr(\top)$ is sufficient for evaluating $\Pr(S \rightsquigarrow v)$.

3.2 Procedure and Complexity

Now we describe the procedure to compute $\Pr(\top)$. We again use the transition of TCs. By considering the pruning described above, we redefine the set of TCs as $\mathcal{N}'_1 \coloneqq \{\Phi_1^v(\emptyset)\}$ and $\mathcal{N}'_i \coloneqq \left(\bigcup_{\Phi \in \mathcal{N}'_{i-1}} \{\mathsf{hi}^v(\Phi), \mathsf{lo}^v(\Phi)\}\right) \setminus \{\top, \bot\}$, i.e., we remove the pruned TCs from $\mathcal{N}_i$. Then, from the definition of $\Pr(\Phi)$ (Eq. (6)), we have

$$\Pr(\Phi) = p_i \cdot \sum_{\Phi' \in \mathcal{N}'_{i-1} : \mathsf{hi}(\Phi') = \Phi} \Pr(\Phi') + (1 - p_i) \cdot \sum_{\Phi' \in \mathcal{N}'_{i-1} : \mathsf{lo}(\Phi') = \Phi} \Pr(\Phi'). \tag{7}$$

To compute $\Pr(\top)$, we can use a similar formula:

$$\Pr(\top) = p_i \cdot \sum_{\Phi' \in \mathcal{N}'_1 \cup \cdots \cup \mathcal{N}'_m : \mathsf{hi}(\Phi') = \top} \Pr(\Phi') + (1 - p_i) \cdot \sum_{\Phi' \in \mathcal{N}'_1 \cup \cdots \cup \mathcal{N}'_m : \mathsf{lo}(\Phi') = \top} \Pr(\Phi'). \tag{8}$$

Starting from $\mathcal{N}'_1 = \{\Phi_1^v(\emptyset)\}$, we can generate $\mathcal{N}'_{i+1}$ from $\mathcal{N}'_i$ recursively using transition functions $\mathsf{hi}^v, \mathsf{lo}^v$. By recording the transition, we can compute every $\Pr(\Phi)$ and $\Pr(\top)$ with (7) and (8).

Algorithm 1 formally presents the procedures. Lines 1–7 recursively generate the successive TCs using transition functions $\mathsf{lo}^v, \mathsf{hi}^v$ while executing $\bot, \top$-pruning. This process can be seen as constructing a diagram of TCs as in Figure 3. Every Φ except for $\bot, \top$ has two outgoing edges heading to $\mathsf{lo}^v(\Phi)$ and $\mathsf{hi}^v(\Phi)$. In Algorithm 1, each Φ has two entries $\Phi.\mathsf{lo}$ and $\Phi.\mathsf{hi}$ that store the pointer to $\mathsf{lo}^v(\Phi)$ and $\mathsf{hi}^v(\Phi)$. Here, $\mathcal{N}'_i$ stores the TCs of the subsets of $E_{<i}$ as a hash map; pruned TCs are not stored. Along this diagram, we can compute every $\Pr(\Phi)$ with simple dynamic programming (DP) in lines 8–13; this simulates the formulas (7) and (8). The array $\mathsf{p}[\Phi]$ for $\Phi \in \mathcal{N}'_i$ is prepared to store $\Pr(\Phi)$. Note that even if an identical TC Φ appears in $\mathcal{N}'_i$ and $\mathcal{N}'_j$ $(i \neq j)$, we distinguish $\Phi \in \mathcal{N}'_i$ and $\Phi \in \mathcal{N}'_j$ in computing $\mathsf{p}[\cdot]$. After computing all $\mathsf{p}[\Phi]$, $\Pr(S \rightsquigarrow v) = \Pr(\top)$ is stored in $\mathsf{p}[\top]$.

Next, we analyze the complexity of this algorithm. For each TC Φ, computing $\mathsf{lo}^v(\Phi)$ and $\mathsf{hi}^v(\Phi)$ takes $O((|W_i| + |W_{i+1}|)^2)$ time; details are in Lemma 4.14 of Section 4.4. If every $\mathcal{N}'_i$ is implemented as a hash map, storing or searching a TC also takes $O(|W_i|^2)$ time. The number of patterns on Φ_i^v can be bounded by $O(2^{|W_i|^2})$; since this was not rigorously proved [15], we prove it in Lemma 4.15 of Section 4.4. The overall complexity can be bounded by $O(\sum_i (|W_i| + |W_{i+1}|)^2 \cdot 2^{|W_i|^2}) = O(m\omega^2 \cdot 2^{\omega^2})$, where $\omega = \max_i |W_i|$.

In fact, ω is related to the pathwidth of a graph. Given a path decomposition of G with width ω_p, we can generate an edge ordering with $\omega \leq \omega_p$ in linear time [9]. Since we can construct a path decomposition in linear time for a graph with bounded pathwidth [2], we have a linear time algorithm for computing $\Pr(S \rightsquigarrow v)$ on a graph with bounded pathwidth. However, computing $\sigma(S)$ requires $O(mn\omega^2 \cdot 2^{\omega^2})$ time by computing $\Pr(S \rightsquigarrow v)$ for $v \in V \setminus S$.

4 Linear-Time Influence Spread Computation

We now describe the proposed algorithm for Problem 2.1. First, we explain the high-level concept of the proposed algorithm. In computing $\sigma(S)$ with the algorithm described in Section 3, different diagrams of TCs are generated for different $v \in V \setminus S$ in computing $\Pr(S \rightsquigarrow v)$. To argue the similarities among these diagrams, we divide each of them into three parts: *upper levels*, *lower levels*, and *middle levels*. More specifically, let $A_i \subseteq V$ be the vertices

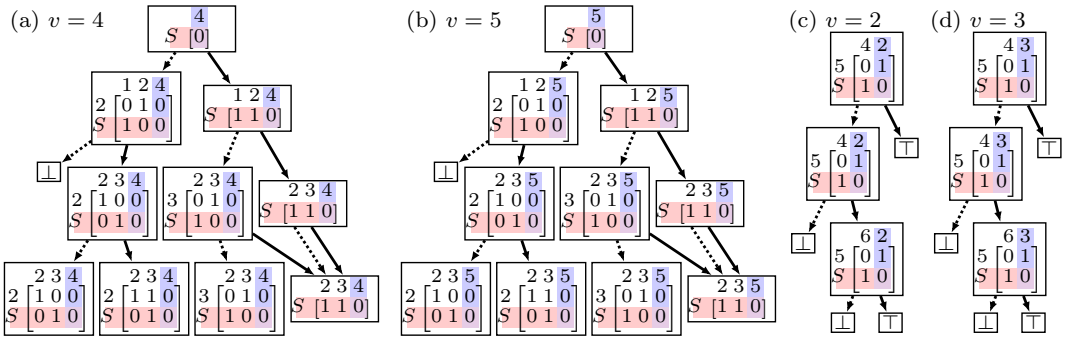

Figure 3 (a)(b) Upper four levels of diagrams constructed from graph in Figure 2a with $v = 4$ and $v = 5$. (c)(d) Parts of lower levels of diagrams constructed from graph in Figure 2a with $v = 2$ and $v = 3$; (c) depicts part below $\Phi_8^2(\{e_2, e_5, e_6\})$ and (d) depicts part below $\Phi_8^3(\{e_1, e_4, e_7\})$. Solid arcs indicate connection from Φ to $\Phi.\mathsf{hi}$, while dashed arcs indicate connection from Φ to $\Phi.\mathsf{lo}$.

that only appear in $E_{<i}$ and do not appear in $E_{\geq i}$, and $B_i \subseteq V$ be those that only appear in $E_{\geq i}$ and do not appear in $E_{<i}$. A_i, B_i, and W_i (Definition 3.1) constitute a partition of V. Moreover, since $E_{<i} \subset E_{<i+1}$ and $E_{\geq i} \supset E_{\geq i+1}$ for $i = 1, \ldots, m$, $V = B_1 \supseteq \cdots \supseteq B_{m+1} = \emptyset$ and $\emptyset = A_1 \subseteq \cdots \subseteq A_{m+1} = V$. Thus, by fixing v, the diagram of TCs $\Phi_i^v(\cdot)$ made from the reachability to v can be divided into three parts according to i: levels i s.t. $v \in B_i$ (upper levels), those s.t. $v \in W_i$ (middle levels), and those s.t. $v \in A_i$ (lower levels).

We first find that upper levels of these diagrams exhibit an identical structure, i.e., diagrams of Φ_i^v and Φ_i^w exhibit an identical structure as long as $v, w \in B_i$ (Observation 4.2); we extract the essential information to build this identical diagram as a *shared transversal configuration* (STC) $\Psi = \Psi_i(E')$. We can also confirm that middle levels of these diagrams were also governed by the STCs (Lemma 4.8). Thus, by considering STCs $\Psi_i(\cdot)$, we can have a similar formula to (5) using the STCs when $v \in B_i \cup W_i$:

$$\Pr(S \rightsquigarrow v) = \sum_{\Psi \in \mathcal{M}_i} \Pr(\Psi) \cdot \Pr(S \rightsquigarrow v \mid \Psi), \tag{9}$$

$$\text{where} \quad \Pr(\Psi) \coloneqq \sum_{E' \subseteq E_{<i} : \Psi_i(E') = \Psi} \Pr_{<i}(E'), \quad \mathcal{M}_i \coloneqq \{\Psi_i(E') \mid E' \subseteq E_{<i}\}.$$

For every v, there exists i such that $v \in B_i \cup W_i$. Thus, for every v, if $\Pr(\Psi)$ and $\Pr(S \rightsquigarrow v \mid \Psi)$ are efficiently computed for all $\Psi = \Psi_i(\cdot)$ with the above i, we can compute $\Pr(S \rightsquigarrow v)$ with (9). Here, since $\Pr(\Psi)$ has a similar recursion as $\Pr(\Phi)$ (Lemma 4.9), all $\Pr(\Psi)$s can be computed in a manner similar to that used in Section 3 by building a diagram of STCs from $i = 1$ to $m + 1$. Since $|\mathcal{M}_i|$ is constant for a graph with bounded pathwidth (Lemma 4.16), it can be performed in $O(m)$ time.

Indeed, when $v \in W_i$ (middle levels), $\Pr(S \rightsquigarrow v \mid \Psi)$ for $\Psi = \Psi_i(\cdot)$ can also be computed efficiently. To compute this, we use the lower levels of the diagram of TCs $\Psi_i^v(\cdot)$. Although naively constructing the diagrams of TCs Φ for every v costs $O(mn)$ time, we find that the lower levels of these diagrams have many identical substructures (Lemma 4.5). By using this, we can build all possible substructures of diagrams that appear in the lower level of these diagrams in $O(m)$ time. In addition, we again use middle levels' equivalence (Lemma 4.8) to compute $\Pr(S \rightsquigarrow v \mid \Psi)$ on middle levels in $O(m)$ time. Here we use the recursion on $\Pr(S \rightsquigarrow v \mid \Psi)$ (Lemmas 4.11 and 4.12).

Eventually, for every v and i where $v \in W_i$, we obtain $\Pr(\Psi)$ and $\Pr(S \rightsquigarrow v \mid \Psi)$ for $\Psi = \Psi_i(\cdot)$. Thus, by choosing i such that $v \in W_i$, we can compute $\Pr(S \rightsquigarrow v)$ with (9). Since (9) contains $O(|\mathcal{M}_i|)$ terms, which is constant for bounded-pathwidth graphs, the

computation of $\Pr(S \rightsquigarrow v)$ for every v costs $O(n)$ time in total. The overall time complexity is bounded by $O(m+n)$ for a graph with bounded pathwidth. Note that, for every v, there exists i such that $v \in W_i$ if v has degree more than 1. For vertices with degree 1, such i might not exist, but we can easily compute $\Pr(S \rightsquigarrow v)$ for such vertices, as described later.

In the following, we first identify the similarities in the previous algorithm [15] for different v in Section 4.1. Then, we derive DP formulas for computing $\Pr(S \rightsquigarrow v)$ in Section 4.2. Finally, we explain the main procedures and the complexity in Sections 4.3 and 4.4.

4.1 Equivalence of Diagrams for Different Vertices

We formally state the equivalence of different diagrams of TCs for upper levels ($v \in B_i$), lower levels ($v \in A_i$), and middle levels ($v \in W_i$).

Upper level equivalence. First, we consider the case $v, w \in B_i$. No edges in $E_{<i}$ are incident to v, meaning that no vertex in W_i can reach v on $(V, E_{<i})$. Furthermore, no vertex in S can reach v on $(V, E_{<i})$. Therefore, for any $E' \subseteq E_{<i}$, $\Phi_i^v(E')_{uv} = 0$ for any u. This means we need to consider the reachability relations among only $W_i \cup \{S\}$, not $W_i \cup \{S, v\}$.

▶ **Definition 4.1.** *Let $\Psi_i(E')$ be a binary matrix indexed by $(W_i \setminus W_i^{S\rightsquigarrow}(E') \cup \{S\}) \times W_i$, where $\Psi_i(E')_{ux} = 1$ if and only if $u \rightsquigarrow_{(V,E')} x$. Here, S is treated as a special (single) row index. We call $\Psi_i(\cdot)$ a* shared transversal configuration *(STC).*

Since $v \in B_i$, we can recover $\Phi_i^v(E')$ from $\Psi_i(E')$ by $\Phi_i^v(E')_{ux} = \Psi_i(E')_{ux}$ for any u as well as $x \neq v$ and $\Phi_i^v(E')_{uv} = 0$ for any u. In other words, the diagram structure is completely determined by the transition of $\Psi_i(\cdot)$ instead of $\Phi_i^v(\cdot)$ when $v \in B_i$. The above argument also holds for the other $w \in B_i$. Therefore, the following observation holds.

▶ **Observation 4.2.** *Consider the diagrams generated using Algorithm 1 with different v, w. If $v, w \in B_i$, the diagrams above any TCs $\Phi_i^v(E')$ and $\Phi_i^w(E')$ are identical. Moreover, we can instead use STC $\Psi_i(\cdot)$ in building the diagram.*

▶ **Example 4.3.** Figures 3a and 3b show the upper four levels of the diagram generated using Algorithm 1 with the graph of Figures 2a and $v = 4$ or $v = 5$. Since $4, 5 \in B_4$, the first four levels are identical, as depicted in these figures.

Lower level equivalence. Next, we consider the case $v, w \in A_i$. Recall that $\Phi_i^v(\cdot)$ captures the reachability among $W_i \cup \{S, v\}$, and $\Phi_i^w(\cdot)$ captures those among $W_i \cup \{S, w\}$. Since $v, w \in A_i$, all edges incident to v and w are in $E_{<i}$, meaning that all incident edges are determined when the edges in $E_{<i}$ are determined. Thus, intuitively, if Φ_i^v and Φ_i^w are "identical," i.e., the reachability relations are identical by substituting v with w, $S \rightsquigarrow v$ and $S \rightsquigarrow w$ can be regarded as equivalent events. We formally define such an equivalence.

▶ **Definition 4.4.** *For $E', F' \subseteq E_{<i}$, consider two TCs $\Phi_i^v(E')$ and $\Phi_i^w(F')$, where $v, w \in A_i$. We say these TCs are* identical *if $\Phi_i^v(E')_{ux} = \Phi_i^w(F')_{ux}$ for any u and $x \neq v, w$ and $\Phi_i^v(E')_{uv} = \Phi_i^w(F')_{uw}$ for any u. This means that $\Phi_i^v(E')$ and $\Phi_i^w(F')$ are identical matrices, except for the column indices being different.*

▶ **Lemma 4.5.** *Suppose $\Phi_i^v(E')$ and $\Phi_i^w(F')$ are identical for $v, w \in A_i$ and $E', F' \subseteq E_{<i}$. Then, for any $H \subseteq E_{\geq i}$, $S \rightsquigarrow_{(V,E' \cup H)} v$ if and only if $S \rightsquigarrow_{(V,F' \cup H)} w$.*

Proof. Fix $H \subseteq E_{\geq i}$ and suppose that $S \rightsquigarrow_{(V,E' \cup H)} v$. There is then a directed walk P from some $s \in S$ to v in $(V, E' \cup H)$. The walk P can be divided into small walks $P = P_0 P_1 P_2 \cdots P_{2k-1} P_{2k}$ satisfying the conditions that P_{2j} consists of only the edges in $E' \subseteq E_{<i}$ and that P_{2j+1} consists of only the edges in $H \subseteq E_{\geq i}$. Let p_j be the contact vertex of P_j and P_{j+1}. Note that P_0 may be empty when $s \in W_i \cup B_i$; in such a case, we set $p_0 = s$. Since p_j $(j \geq 1)$ is incident to both edges in $E' \subseteq E_{<i}$ and those in $H \subseteq E_{\geq i}$, $p_j \in W_i$. If P_0 is nonempty, p_0 is also in W_i. We can also assume that $p_1, p_2, \ldots, p_{2k-2} \notin W_i^{S \rightsquigarrow}(E')$; otherwise, we can take a directed walk $P^\star$ from some $s \in S$ to p_j within (V, E') and consider a walk $P^\star P_{j+1} \cdots P_{2k}$ instead. With a similar argument, we can assume that $p_1, p_2, \ldots, p_{2k-2} \notin W_i^{\rightsquigarrow v}(E')$. Now, we constitute a directed walk P' from some $s' \in S$ to w in $(V, F' \cup H)$. First, when P_0 is not empty, $\Phi_i^v(E')_{Sp_0} = 1$. Since $\Phi_i^v(E')$ and $\Phi_i^w(F')$ are identical, $\Phi_i^w(F')_{Sp_0} = 1$; thus, there is a directed walk P'_0 from some $s' \in S$ to p_0 in (V, F'). This also holds when P_0 is empty; in such a case, $s' = s = p_0$ and P'_0 is also empty. Next, for $1 \leq j \leq k-1$, P_{2j}'s existence and identity indicate $\Phi_i^v(E')_{p_{2j-1}p_{2j}} = 1 = \Phi_i^w(F')_{p_{2j-1}p_{2j}}$, meaning that there is a directed walk P'_{2j} from p_{2j-1} to p_{2j} in (V, F'). Finally, P_{2k}'s existence and identity indicate $\Phi_i^v(E')_{p_{2k-1}v} = 1 = \Phi_i^w(F')_{p_{2k-1}w}$, meaning that there is a directed walk P'_{2k} from p_{2k-1} to w in (V, F'). Now $P' = P'_0 P_1 P'_2 \cdots P_{2k-1} P'_{2k}$ is a directed walk from $s' \in S$ to w in $(V, F' \cup H)$, indicating $S \rightsquigarrow_{(V,F' \cup H)} w$.

In the same manner, we can show the reverse direction. Thus, the lemma holds. ◀

This indicates that $\Pr(S \rightsquigarrow v \mid \Phi_i^v) = \Pr(S \rightsquigarrow w \mid \Phi_i^w)$ when Φ_i^v and Φ_i^w are identical.

▶ **Observation 4.6.** *Consider the diagrams generated using Algorithm 1 with different v, w. If $v, w \in A_i$ and identical TCs Φ_i^v, Φ_i^w appear in each diagram, the diagram structures below Φ_i^v and below Φ_i^w are identical.*

▶ **Example 4.7.** Figures 3c and 3d show parts of the last four levels of the diagrams below $\Phi_8^2(\{e_2, e_5, e_6\})$ $(v = 2)$ and below $\Phi_8^3(\{e_1, e_4, e_7\})$ $(v = 3)$. Since $2, 3 \in A_8$ and these TCs are identical, these diagram structures are also identical.

Middle level equivalence. Finally, we examine the case $v, w \in W_i$. In this case, the TC $\Phi_i^v(\cdot)$ captures the reachability among $W_i \cup \{S, v\} = W_i \cup \{S\}$. In addition, the STC $\Psi_i(\cdot)$ also captures the reachability among $W_i \cup \{S\}$. Thus, we can recover $\Phi_i^v(E')$ from $\Psi_i(E')$ by removing the columns corresponding to $W_i^{\rightsquigarrow v}(E')$. In other words, if $\Psi_i(E') = \Psi_i(F')$, then $E', F' \subseteq E_{<i}$ satisfy condition (#).

For $\Psi = \Psi_i(E')$, let $W_i^{S \rightsquigarrow}(\Psi)$ be the set of vertices in W_i that can be reached from S under Ψ, which is determined by the missing row indices of Ψ. Consider a graph $G_i(\Psi) \coloneqq (W_i \setminus W_i^{S \rightsquigarrow}(\Psi), E_i(\Psi))$, where $E_i(\Psi) \coloneqq \{(u, x) \mid u, x \in W_i \setminus W_i^{S \rightsquigarrow}(\Psi), \Psi_{ux} = 1\}$. In other words, when $\Psi = \Psi_i(E')$, $G_i(\Psi)$ is the subgraph of the transitive closure of (V, E') induced by $W_i \setminus W_i^{S \rightsquigarrow}(\Psi)$. Then, we can prove the following.

▶ **Lemma 4.8.** *Suppose $\Psi_i(E') = \Psi_i(F') =: \Psi$ for $E', F' \subseteq E_{<i}$ and $v, w \in W_i \setminus W_i^{S \rightsquigarrow}(E')$ are in the same SCC of $G_i(\Psi)$. Then, for any $H \subseteq E_{\geq i}$, $S \rightsquigarrow_{(V,E' \cup H)} v$ if and only if $S \rightsquigarrow_{(V,F' \cup H)} w$.*

Proof. Fix $H \subseteq E_{\geq i}$. We can prove the equivalence of $S \rightsquigarrow_{(V,E' \cup H)} v$ and $S \rightsquigarrow_{(V,F' \cup H)} v$ from $\Psi_i(E') = \Psi_i(F')$ with almost the same argument as that in the proof of Lemma 4.5. Since v and w are in the same SCC of $G_i(\Psi)$, $v \rightsquigarrow_{(V,F')} w$ and $w \rightsquigarrow_{(V,F')} v$. Consequently, $S \rightsquigarrow_{(V,F' \cup H)} v$ indicates $S \rightsquigarrow_{(V,F' \cup H)} w$ and vice versa. This concludes $S \rightsquigarrow_{(V,E' \cup H)} v \iff S \rightsquigarrow_{(V,F' \cup H)} w$. ◀

4.2 New Decomposition and DP Formulas

Decomposition formula. Using the equivalences described above, we derive a formula for $\Pr(S \rightsquigarrow v)$ using the STCs. Since we confirmed that $\Phi_i^v(E')$ can be recovered from $\Psi_i(E')$ when $v \in A_i \cup W_i$, we can derive the decomposition formula (9). If $v \in W_i$, it can be further transformed. Based on Lemma 4.8, we can rewrite $\Pr(S \rightsquigarrow v \mid \Psi)$ as $\Pr(S \rightsquigarrow C_\Psi(v) \mid \Psi)$, where $C_\Psi(v)$ is the SCC of $G_i(\Psi)$ containing v when $v \notin W_i^{S\rightsquigarrow}(\Psi)$. If $v \in W_i^{S\rightsquigarrow}(\Psi)$, we immediately have $\Pr(S \rightsquigarrow v \mid \Psi) = 1$. Now (9) can be further transformed into

$$\Pr(S \rightsquigarrow v) = \sum_{\Psi \in \mathcal{M}_i} \begin{cases} \Pr(\Psi) & (v \in W_i^{S\rightsquigarrow}(\Psi)) \\ \Pr(\Psi) \cdot \Pr(S \rightsquigarrow C_\Psi(v) \mid \Psi) & (v \in W_i \setminus W_i^{S\rightsquigarrow}(\Psi)) \end{cases}. \tag{10}$$

If we can obtain $\Pr(\Psi)$ and $\Pr(S \rightsquigarrow C \mid \Psi)$ for every STC Ψ and every SCC C of $G_i(\Psi)$, we can compute every $\Pr(S \rightsquigarrow v)$ by choosing i such that $v \in W_i$ and then use (10). Note that if all vertices have degree not less than 2, there must exist index i such that $v \in W_i$, since G contains no self-loops. The $\Pr(S \rightsquigarrow v)$ value for degree 1 vertices can be easily obtained; details are given in Section 4.3.

DP formulas. Since TCs can be recovered from STCs and admit transition functions $\mathsf{hi}^v, \mathsf{lo}^v$, STCs also admit transition functions lo and hi satisfying that, for any $E' \subseteq E_{<i}$, $\mathsf{lo}(\Psi_i(E')) = \Psi_{i+1}(E')$ and $\mathsf{hi}(\Psi_i(E')) = \Psi_{i+1}(E' \cup \{e_i\})$. Here, we set $\mathsf{f}(\Psi) = \bot$ for $\mathsf{f} \in \{\mathsf{hi}, \mathsf{lo}\}$ when $i \geq i_S$ and $\mathsf{f}(\Psi)_{Su} = 0$ for any u, which confirms $\Pr(S \rightsquigarrow w \mid \mathsf{f}(\Psi)) = 0$ for any further vertices $w \in W_{i+1} \cup B_{i+1}$ ($\bot$-pruning). Now $\Pr(\Psi)$ can be computed in the same manner as described in Section 3: From the definition of $\Pr(\Psi)$ (Eq. (9)), we have the following.

▶ **Lemma 4.9.** *For $\Psi = \Psi_i(E')$ for some $E' \subseteq E_{<i}$, we have*

$$\Pr(\Psi) = p_i \cdot \sum_{\Psi' \in \mathcal{M}'_{i-1} : \mathsf{hi}(\Psi') = \Psi} \Pr(\Psi') + (1 - p_i) \cdot \sum_{\Psi' \in \mathcal{M}'_{i-1} : \mathsf{lo}(\Psi') = \Psi} \Pr(\Psi'), \tag{11}$$

where $\mathcal{M}'_1 := \{\Psi_1(\emptyset)\}$ and $\mathcal{M}'_i := \left(\bigcup_{\Psi \in \mathcal{M}'_{i-1}} \{\mathsf{hi}(\Psi), \mathsf{lo}(\Psi)\}\right) \setminus \{\bot\}$, i.e., we remove pruned STCs from $\mathcal{M}_i$.

It remains to be explained how to compute $\Pr(S \rightsquigarrow C \mid \Psi)$ for an SCC C of $G_i(\Psi)$. If there exists $u \in C$ such that $u \in W_{i+1}$, we can decompose it using the case analysis of e_i: If e_i is present, $\Pr(S \rightsquigarrow C \mid \Psi)$ equals $\Pr(S \rightsquigarrow C' \mid \mathsf{hi}(\Psi))$, where C' is the SCC of $G_{i+1}(\mathsf{hi}(\Psi))$ containing u. If e_i is absent, $\Pr(S \rightsquigarrow C \mid \Psi)$ equals $\Pr(S \rightsquigarrow C'' \mid \mathsf{lo}(\Psi))$, where C'' is the SCC of $G_{i+1}(\mathsf{lo}(\Psi))$ containing u. Here, the relation between C and C', C'' can be seen as the transition from C to successive SCCs C', C''. By considering the pruned cases, we formally define the successive SCCs as follows.

▶ **Definition 4.10.** *For SCC C of $G_i(\Psi)$, we define* successive SCC $\mathsf{f}(C)$ *for $\mathsf{f} \in \{\mathsf{lo}, \mathsf{hi}\}$ as follows. (i) If $\mathsf{f} = \mathsf{hi}$, and $e_i = (u, v)$ with $u \in W_i^{S\rightsquigarrow}(\Psi) \cup S$ and $v \in C$, $\mathsf{hi}(C) := \top$, meaning that C can be reached from S by determining the presence of e_i. (ii) If (i) does not hold and $\mathsf{f}(\Psi) = \bot$, $\mathsf{f}(C) := \bot$, meaning that C cannot be reached from S. (iii) If neither (i) nor (ii) holds and there exists $u \in C$ such that $u \in W_{i+1}$, $\mathsf{f}(C) := C_{\mathsf{f}(\Psi)}(u)$. (iv) Otherwise, $\mathsf{f}(C) := \emptyset$, meaning that there is no successive SCC.*

▶ **Lemma 4.11.** *If there exists $u \in C$ such that $u \in W_{i+1}$, i.e., successive SCCs exist, we can decompose $\Pr(S \rightsquigarrow C \mid \Psi)$ for $\Psi = \Psi_i(E')$ and SCC C of $G_i(\Psi)$ as follows:*

$$\Pr(S \rightsquigarrow C \mid \Psi) = p_{e_i} \cdot \Pr(S \rightsquigarrow \mathsf{hi}(C) \mid \mathsf{hi}(\Psi)) + (1 - p_{e_i}) \cdot \Pr(S \rightsquigarrow \mathsf{lo}(C) \mid \mathsf{lo}(\Psi)). \tag{12}$$

Note that if $u \in W_{i+1}^{S\rightsquigarrow}(\mathsf{hi}(\Psi))$, since $\Pr(S \rightsquigarrow u \mid \mathsf{hi}(\Psi)) = 1$, we replace $\Pr(S \rightsquigarrow C_{\mathsf{hi}(\Psi)}(u) \mid \mathsf{hi}(\Psi))$ in the above formula with 1. This replacement is reflected in case (i) of Definition 4.10.

What can be done when such a vertex u does not exist? In this case, we attempt to use the original TC. We observe in Section 4.1 that if $u \in W_i$, we can recover $\Phi = \Phi_i^u(E')$ from $\Psi_i(E')$ with an appropriate transformation. Thus, by fixing some $u \in C$ and letting Φ^u be a TC recovered from STC Ψ, we have the following case analysis: If e_i is present, $\Pr(S \rightsquigarrow C \mid \Psi) = \Pr(S \rightsquigarrow u \mid \Phi^u)$ equals $\Pr(S \rightsquigarrow u \mid \mathsf{hi}^u(\Phi^u))$. If e_i is absent, $\Pr(S \rightsquigarrow C \mid \Psi)$ equals $\Pr(S \rightsquigarrow u \mid \mathsf{lo}^u(\Phi^u))$. This leads to the following decomposition.

▶ **Lemma 4.12.** *For $\Psi = \Psi_i(E')$ and SCC C of $G_i(\Psi)$, suppose that successive SCCs do not exist. Let $u \in C$ and Φ^u be a TC recovered from STC Ψ. Then,*

$$\Pr(S \rightsquigarrow C \mid \Psi) = p_{e_i} \cdot \Pr(S \rightsquigarrow u \mid \mathsf{hi}^u(\Phi^u)) + (1 - p_{e_i}) \cdot \Pr(S \rightsquigarrow u \mid \mathsf{lo}^u(\Phi^u)). \tag{13}$$

To compute the terms in the right-hand-side of (13), we can use the previous algorithm in Section 3. However, naively computing these terms for every $u \in V$ requires $O(mn\omega_p^2 \cdot 2^{\omega_p^2})$ time, since this amounts to constructing a diagram of the previous algorithm for every u.

To overcome this, we use Lemma 4.5. Since we only need to compute $\Pr(S \rightsquigarrow v \mid \Phi^v)$ for the case $v \in A_i$, we can share identical TCs, as in Observation 4.6. Accordingly, we do not need to construct different diagrams for every v to compute $\Pr(S \rightsquigarrow v \mid \Phi^v)$, that is, if identical TCs Φ^v and Φ^w from different v, w appear, we share them. This enables us to compute all $\Pr(S \rightsquigarrow v \mid \Phi^v)$s in linear time.

4.3 Procedure

We formally describe the proposed method in Algorithm 2. Lines 1–6 construct the diagram of STCs Ψ. $\Psi.\mathsf{f}$ stores a pointer to $\mathsf{f}(\Psi)$. We consider the successive SCCs in lines 7–11. If a successive SCC of C exists, $C.\mathsf{f}$ stores the pointer to it. Otherwise, following the approach of Section 4.2, we recover the original TC Φ^u, where $u \in C$, and then carry out transition lo^u or hi^u on it. $\Phi^u(\Psi)$ recovers the original TC by removing every column x satisfying $\Psi_{xu} = 1$ from Ψ, that is, we remove columns whose indices can reach u. If a pruning occurs by considering $\Phi = \mathsf{f}^u(\Phi^u(\Psi))$, $C.\mathsf{f}$ stores $\bot$ or $\top$. Otherwise, $C.\mathsf{f}$ stores a pointer to $\mathsf{change}^t(\Phi)$, where $\mathsf{change}^t(\cdot)$ changes the column index u to t. Here, t is an imaginary vertex that is assumed to be contained within A_i. Since Observation 4.6 confirms that the transitions of identical TCs are identical, we standardize the column index u by changing it to t. We build a diagram of $\Phi_i^t(\cdot)$ in Lines 12–15.

Lines 17–19 compute $\mathsf{p}[\Psi] = \Pr(\Psi)$ for every $\Psi \in \mathcal{M}_i'$ in the same manner as Algorithm 1, and lines 20–22 compute $\mathsf{q}[\Phi] = \Pr(S \rightsquigarrow t \mid \Phi)$ for every $\Phi \in \mathcal{N}_i'$ in a bottom-up manner. Lines 23–26 compute $\mathsf{r}[\Psi, C] = \Pr(S \rightsquigarrow C \mid \Psi)$ for every Ψ and every SCC C of $G_i(\Psi)$. Here, $\langle c \ ? \ x : y \rangle$ stands for a conditional operator that returns x if c is satisfied or y otherwise. When $C.\mathsf{f}$ is either an original TC Φ^t or $\bot, \top$, we use the q value; otherwise, we use the r value. Finally, lines 27–31 compute every $\mathsf{res}[v] = \Pr(S \rightsquigarrow v)$ using (10).

Here, we confirm the correctness of Algorithm 2.

▶ **Lemma 4.13.** *After executing Algorithm 2, $\mathsf{res}[v]$ equals $\Pr(S \rightsquigarrow v)$.*

Proof. First, we confirm $\mathsf{p}[\Psi] = \Pr(\Psi)$. Starting from $\mathsf{p}[\Psi_1(\emptyset)] = \Pr(\Psi_1(\emptyset)) = 1$ for $\Psi_1(\emptyset) \in \mathcal{M}_1'$, lines 17–19 compute every $\Pr(\Psi)$ value for $\mathcal{M}_2', \ldots, \mathcal{M}_m'$ according to (11).

Second, we confirm $\mathsf{q}[\Phi] = \Pr(S \rightsquigarrow t \mid \Phi)$, but this is much simpler; with the case analysis of whether e_i is present or absent, we have

$$\Pr(S \rightsquigarrow t \mid \Phi) = p_i \cdot \Pr(S \rightsquigarrow t \mid \mathsf{hi}^t(\Phi)) + (1 - p_i) \cdot \Pr(S \rightsquigarrow t \mid \mathsf{lo}^t(\Phi)). \tag{14}$$

Algorithm 2 Proposed algorithm for computing $\Pr(S \rightsquigarrow v)$ for every v.

```
1  M'_1 ← {Ψ_1(∅)}, M'_i ← {} (i ≥ 2), N'_i ← {} (i ≥ 1)
2  for i ← 1 to m do                                          // Diagram construction for Ψ
3      foreach Ψ ∈ M'_i do
4          foreach f ∈ {lo, hi} do
5              if f(Ψ) = ⊥ then Ψ.f ← ⊥
6              else M'_{i+1} ← M'_{i+1} ∪ {f(Ψ)}, Ψ.f ← f(Ψ) ∈ M'_{i+1}
7              foreach C: SCC of G_i(Ψ) do                    // Compute successive SCCs
8                  if f(C) ≠ ∅ then C.f ← f(C)                // Including the cases f(C) = ⊥, ⊤
9                  else                                       // Choose u ∈ C arbitrarily
10                     if Φ ← f^u(Φ^u(Ψ)) = ⊥ or ⊤ then C.f ← ⊥ or ⊤
11                 else N'_{i+1} ← N'_{i+1} ∪ {change^t(Φ)}, C.f ← change^t(Φ) ∈ N'_{i+1}
12     foreach Φ ∈ N'_i do                                    // Diagram construction for Φ^t
13         foreach f ∈ {lo, hi} do
14             if f^t(Φ) = ⊥ or ⊤ then Φ.f ← ⊥ or ⊤
15             else N'_{i+1} ← N'_{i+1} ∪ {f^t(Φ)}, Φ.f ← f^t(Φ) ∈ N'_{i+1}
16 p[Ψ] ← 1 for Ψ ∈ M'_1, p[Ψ] = 0 for Ψ ∈ M'_i (i ≥ 2), q[⊥] ← 0, q[⊤] ← 1
17 for i ← 1 to m do                                          // Top-down DP
18     foreach Ψ ∈ M'_i do                                    // p[Ψ] stores Pr(Ψ)
19         p[Ψ.lo] += (1 − p_{e_i}) · p[Ψ], p[Ψ.hi] += p_{e_i} · p[Ψ]
20 for i ← m to 1 do                                          // Bottom-up DP
21     foreach Φ ∈ N'_i do                                    // q[Φ] stores Pr(S ⇝ t | Φ)
22         q[Φ] ← (1 − p_{e_i}) · q[Φ.lo] + p_{e_i} · q[Φ.hi]
23     foreach Ψ ∈ M'_i do
24         foreach C: SCC of G_i(Ψ) do                        // r[Ψ, C] stores Pr(S ⇝ C | Ψ)
25             r[Ψ, C] ← (1 − p_{e_i}) · ⟨lo(C) ∈ {⊤, ⊥, ∅} ? q[C.lo] : r[Ψ.lo, C.lo]⟩
26                       + p_{e_i} · ⟨hi(C) ∈ {⊤, ⊥, ∅} ? q[C.hi] : r[Ψ.hi, C.hi]⟩
27 foreach v ∈ V \ S do                                       // Compute Pr(S ⇝ v) by using Eq. (10)
28     res[v] ← 0                                             // res[v] stores Pr(S ⇝ v)
29     foreach Ψ ∈ M'_{i*} do                                 // Choose i* arbitrarily s.t. v ∈ W_{i*}
30         res[v] += p[Ψ] · ⟨v ∈ W_{i*}^{S⇝}(Ψ) ? 1 : r[Ψ, C_Ψ(v)]⟩
31 return res
```

Starting from $\mathsf{q}[\bot] = \Pr(S \rightsquigarrow t \mid \bot) = 0$ and $\mathsf{q}[\top] = \Pr(S \rightsquigarrow t \mid \top) = 1$, lines 20–22 compute every $\Pr(S \rightsquigarrow t \mid \Phi)$ value for $\mathcal{N}'_m, \mathcal{N}'_{m-1}, \ldots$ according to (14).

Third, we confirm $\mathsf{r}[\Psi, C] = \Pr(S \rightsquigarrow C \mid \Psi)$. If there exists $u \in C$ such that $u \in W_{i+1}$, i.e., a successive SCC exists, we can apply (12). Otherwise (i.e., a successive SCC does not exist), we can apply (13). Here, without considering the pruning, $C.\mathsf{f}$ stores a pointer to $\mathsf{f}(C)$ if a successive SCC exists or, otherwise, to $\mathsf{change}^t(\mathsf{lo}^u(\Phi^u))$. Consequently, lines 23–26 correctly compute $\mathsf{r}[\Psi, C]$ according to (12) if a successive SCC exists or to (13) otherwise. For the latter case, we use Lemma 4.5; i.e., $\Pr(S \rightsquigarrow u \mid \mathsf{lo}^u(\Phi^u)) = \Pr(S \rightsquigarrow t \mid \mathsf{change}^t(\mathsf{lo}^u(\Phi^u))) = \mathsf{q}[\mathsf{change}^t(\mathsf{lo}^u(\Phi^u))]$.

Finally, by choosing $i^\star$ such that $v \in W_{i^\star}$, $\mathsf{res}[v] = \Pr(S \rightsquigarrow v)$ is correctly computed using (10) through lines 27–30. ◀

Treatment of degree 1 vertices. We finally describe how to deal with degree 1 vertices. By obtaining the $\Pr(S \rightsquigarrow v)$ values for all vertices with degree not less than 2, we can easily obtain the $\Pr(S \rightsquigarrow v)$ values for degree 1 vertices. Let $v \in V \setminus S$ be a vertex whose degree is 1. Let e_v be the unique edge incident to v. If $e_v = (v, w)$ for some $w \in V$, i.e., v is the tail of e_v, we have $\Pr(S \rightsquigarrow v) = 0$ because v has no incoming edges. Otherwise (i.e., v is the head of e_v), we have $\Pr(S \rightsquigarrow v) = p_{e_v} \cdot \Pr(S \rightsquigarrow w)$ because every path from some $s \in S$ to v must go through w, thus e_v. Since G is connected and thus w has degree at least 2 except for the trivial case where e_v is the only edge of G, we can obtain $\Pr(S \rightsquigarrow v)$ for v whose degree is 1 from the value of $\Pr(S \rightsquigarrow w)$.

4.4 Complexity

To analyze the complexity, we again use $\omega = \max_i |W_i|$. First, we consider the diagram construction. The key lemma is as follows.

▶ **Lemma 4.14.** *Given $\Phi = \Phi_i^v(E')$ for some $E' \subseteq E_{<i}$, we can compute the transitions $\mathsf{lo}^v(\Phi)$ and $\mathsf{hi}^v(\Phi)$ in $O((|W_i| + |W_{i+1}|)^2)$ time. Similarly, given $\Psi = \Psi_i(E')$ for some $E' \subseteq E_{<i}$, we can also compute $\mathsf{lo}(\Psi)$ and $\mathsf{hi}(\Psi)$ in $O((|W_i| + |W_{i+1}|)^2)$ time.*

The full proof is deferred to the full version; here, we describe the ideas for computing transitions. For TC Φ, we build a graph $G^\star = (W_i \cup \{S, v\}, E^\star)$, where S is regarded as an individual special vertex and $E^\star = \{(u, w) \mid \Phi_{uw} = 1\}$. By adding and removing edges and vertices according to the transition and computing the transitive closure, we can obtain the updated graph, whose adjacency matrix constitutes $\mathsf{hi}(\Phi)$ or $\mathsf{lo}(\Phi)$. The transition of STC Ψ can be computed in very similar way.

Similar to Φ_i^v, we can compute the transition lo, hi of Ψ in $O(\omega^2)$ by recomputing the transitive closure. Note that we can also compute the SCCs and their successive SCCs within this process, as described in the proof of Lemma 4.14. Thus, the total time for diagram construction is bounded by $O(\omega^2) \cdot O(\sum_i (|\mathcal{M}_i| + |\mathcal{N}_i|))$ time. Here, we can bound the number of possible patterns on Φ_i^v and Ψ_i, i.e., $|\mathcal{N}_i|$ and $|\mathcal{M}_i|$, as follows. Since the proofs of Lemmas 4.15 and 4.16 are much similar; the proof of Lemma 4.16 is put on the full version.

▶ **Lemma 4.15.** *$|\mathcal{M}_i| = |\{\Phi_i^v(E') \mid E' \subseteq E_{<i}\}|$ is bounded by $O(2^{|W_i|^2})$.*

Proof. If we fix $W_i^{S\rightsquigarrow}(E')$ and $W_i^{\rightsquigarrow v}(E')$, some entries of $\Phi_i^v(E')$ are determined as follows: $\Phi_i^v(E')_{Sw} = 1$ if $w \in W_i^{S\rightsquigarrow}(E')$ or 0 if $w \in W_i \setminus W_i^{S\rightsquigarrow}(E')$, and $\Phi_i^v(E')_{uv} = 1$ if $u \in W_i^{\rightsquigarrow v}(E')$ or 0 if $u \in W_i \setminus W_i^{\rightsquigarrow v}(E')$. Therefore, the possible patterns on $\Phi_i^v(E')$ is bounded by $O(2^{(|W_i| - |W_i^{S\rightsquigarrow}(E')|)(|W_i| - |W_i^{\rightsquigarrow v}(E')|)})$ because $\Phi_i^v(E')$ is a binary matrix of size $(|W_i| - |W_i^{S\rightsquigarrow}(E')| + 1) \times (|W_i| - |W_i^{\rightsquigarrow v}(E')| + 1)$, and the entries of one row and one column are already determined.

We conduct a case analysis of $k = |W_i^{S\rightsquigarrow}(E')| + |W_i^{\rightsquigarrow v}(E')|$ when fixing $W_i^{S\rightsquigarrow}(E')$ and $W_i^{\rightsquigarrow v}(E')$. When $k = 0$, the bound is $O(2^{|W_i|^2})$. When $k = 1$, the bound is $O(2^{|W_i|^2 - |W_i|})$, and the number of patterns on $(W_i^{S\rightsquigarrow}(E'), W_i^{\rightsquigarrow v}(E'))$ is bounded by $O(|W_i|)$. When $k \geq 2$, since $(|W_i| - |W_i^{S\rightsquigarrow}(E')|)(|W_i| - |W_i^{\rightsquigarrow v}(E')|) \leq (|W_i| - 1)(|W_i| - 1) = |W_i|^2 - 2|W_i| + 1$ (supposing that $|W_i| \geq 3$), the number of possible patterns is bounded by $O(2^{|W_i|^2 - 2|W_i|})$. The number of possible patterns on $(W_i^{S\rightsquigarrow}(E'), W_i^{\rightsquigarrow v}(E'))$ is bounded by $2^{|W_i|} \cdot 2^{|W_i|}$ because $W_i^{S\rightsquigarrow}, W_i^{\rightsquigarrow v} \subseteq W_i$. Thus, the total number of patterns on $\Phi_i^v(E')$ is bounded by $O(2^{|W_i|^2}) + O(|W_i| \cdot 2^{|W_i|^2 - |W_i|}) + O(2^{2|W_i|}) \cdot O(2^{|W_i|^2 - 2|W_i|}) = O(2^{|W_i|^2}) + O(2^{|W_i|^2 + \log|W_i| - |W_i|}) + O(2^{|W_i|^2}) = O(2^{|W_i|^2})$. ◀

▶ **Lemma 4.16.** *$|\mathcal{N}_i| = |\{\Psi_i(E') \mid E' \subseteq E_{<i}\}|$ is bounded by $O(2^{|W_i|^2})$.*

Therefore, $|\mathcal{M}_i| = |\mathcal{N}_i| = O(2^{\omega^2})$, and the total time complexity for diagram construction can be bounded by $O(m\omega^2 \cdot 2^{\omega^2})$.

Next, we consider the DP. Each value can be computed in constant time. The number of values to compute for Ψ_i and Φ_i^v is bounded by $O(|\mathcal{M}_i| + |W_i||\mathcal{M}_i| + |\mathcal{N}_i|) = O(\omega \cdot 2^{\omega^2} + 2^{\omega^2}) = O(\omega \cdot 2^{\omega^2})$. Thus, the total complexity is $O(m\omega \cdot 2^{\omega^2})$.

Finally, computing each $\Pr(S \rightsquigarrow v)$ using (10) requires $O(|\mathcal{M}_i|) = O(2^{\omega^2})$ time. Thus, it needs $O(n \cdot 2^{\omega^2})$ time in total for all $\Pr(S \rightsquigarrow v)$ values.

▶ **Proposition 4.17.** *Given directed graph G, seed set S, each edge's probability p_e of presence, and the edge ordering $e_1, \ldots, e_m$, the probability $\Pr(S \rightsquigarrow v)$ for every vertex v can be computed in $O((m + n) \cdot \omega^2 \cdot 2^{\omega^2})$ time in total.*

Since we can generate an edge ordering with $\omega \leq \omega_p$ from a path decomposition of width ω_p [9] in linear time, we achieve Theorem 1.2. Moreover, with the linear-time path-decomposition algorithm for a bounded-pathwidth graph [2], we have the following.

▶ **Theorem 4.18.** *Given directed graph G, seed set S, and each edge's probability p_e of presence, suppose that G's pathwidth is bounded by a constant. Then, $\Pr(S \rightsquigarrow v)$ for every vertex v can be computed in $O(m+n)$ time in total.*

The previous algorithm [15] requires $O(m\omega^2 \cdot 2^{\omega^2})$ time for computing $\Pr(S \rightsquigarrow v)$, as discussed in Section 3. Thus, it takes $O(nm\omega^2 \cdot 2^{\omega^2})$ time for computing $\sigma(S)$. In contrast, the proposed algorithm takes $O((n+m)\omega^2 \cdot 2^{\omega^2})$ time. Therefore, in terms of time complexity, our algorithm reduces the dependence on the graph size from $O(mn)$ to $O(m+n)$ while the dependence on $\omega = \max_i |W_i|$ is kept the same.

5 Discussion

Here, we point out the differences between the proposed algorithm and the existing algorithm that computes the probability of connection for undirected graphs [16]. These algorithms basically share the same idea: In computing the probability of reachability (or connectivity in the undirected graphs) of every vertex, the previous top-down style algorithms, like that in Section 3, generate similar diagrams for different vertices v, so they attempt to share equivalent structures. The algorithm for undirected graphs uses connected components (CCs) among frontier vertices to efficiently compute the conditional probabilities, similar to $\Pr(S \rightsquigarrow C \mid \Psi)$ in our algorithm. The difference comes from the computation of conditional probability when a successive SCC or CC does not exist. In undirected graphs, if a successive CC does not exist, the current CC is disconnected from any further vertex, so we immediately have the conditional probability 0 (or 1 in an exceptional case). In contrast, in directed graphs, even if a successive SCC does not exist, the current SCC may still reach or be reached from further vertices because it can reach or be reached from the frontier vertices. This makes the computation of the conditional probabilities difficult because we cannot immediately determine them when no successive SCC exists. We resolve this issue by focusing on the equivalence of lower levels of diagrams (Lemma 4.5). This enables us to compute the conditional probabilities using a constant number of diagrams (STCs Ψ and TCs Φ^t of imaginary vertex t) without breaking the linear complexity.

6 Conclusion

We proposed an algorithm for computing the influence spread under IC model exactly in $O((m+n)\omega_p^2 \cdot 2^{\omega_p^2})$ time. For graphs with bounded pathwidth ω_p, we improved the complexity from $O(mn)$ to $O(m+n)$, linear in the graph size, while the dependence on the pathwidth ω_p is kept the same.

Finally, we mention some future directions. First, a more efficient exact evaluation of influence spread, suitable for the greedy algorithm [10], should be investigated. Chen et al. [4] developed an algorithm that simultaneously approximates $\sigma(S \cup \{v\}) - \sigma(S)$ for all $v \in V \setminus S$, which corresponds to one iteration of the greedy algorithm. Nakamura et al. [17] developed an algorithm for undirected graphs that exactly and simultaneously computes the expected number of connected vertices for all vertices. This algorithm is promising for adoption with directed graphs as a way to simultaneously compute $\sigma(S \cup \{v\})$ for all $v \in V \setminus S$ exactly. Second, we should investigate whether a linear time algorithm for computing $\sigma(S)$ is possible for graphs with bounded *treewidth*. A roadmap for achieving this may be to fit

the diagram construction method based on tree decomposition [1] to the computation of influence spread and then observe the equivalences. However, such an improvement may not be straightforward due to the complicated nature of these algorithms.

References

1 Antoine Amarilli, Pierre Bourhis, Louis Jachiet, and Stefan Mengel. A circuit-based approach to efficient enumeration. In *Proc. of the 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, pages 111:1–111:15, 2017. doi:10.4230/LIPIcs.ICALP.2017.111.

2 Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996. doi:10.1137/S0097539793251219.

3 Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proc. of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2010)*, pages 1029–1038, 2010. doi:10.1145/1835804.1835934.

4 Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2009)*, pages 199–208, 2009. doi:10.1145/1557019.1557047.

5 Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proc. of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2001)*, pages 57–66, 2001. doi:10.1145/502512.502525.

6 Jacob Goldenberg, Barak Libai, and Eitan Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 12:211–223, 2001. doi:10.1023/A:1011122126881.

7 Amit Goyal, Wei Lu, and Laks V.S. Lakshmanan. CELF++: optimizing the greedy algorithm for influence maximization in social networks. In *Proc. of the 20th International Conference Companion on World Wide Web (WWW 2011)*, pages 47–48, 2011. doi:10.1145/1963192.1963217.

8 Gary Hardy, Corinne Lucet, and Nikolaos Limnios. K-terminal network reliability measures with binary decision diagrams. *IEEE Transactions on Reliability*, 56(3):506–515, 2007. doi:10.1109/TR.2007.898572.

9 Yuma Inoue and Shin-ichi Minato. Acceleration of ZDD construction for subgraph enumeration via pathwidth optimization. Technical Report TCS-TR-A-16-80, Division of Computer Science, Hokkaido University, 2016.

10 David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2003)*, pages 137–146, 2003. doi:10.1145/956750.956769.

11 Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proc. of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2007)*, pages 420–429, 2007. doi:10.1145/1281192.1281239.

12 Yuchen Li, Ju Fan, Yanhao Wang, and Kian-Lee Tan. Influence maximization on social graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 30(10):1852–1872, 2018. doi:10.1109/TKDE.2018.2807843.

13 Yuchen Li, Dongxiang Zhang, and Kian-Lee Tan. Real-time targeted influence maximization for online advertisements. *Proc. of the VLDB Endowment*, pages 1070–1081, 2015. doi:10.14778/2794367.2794376.

14 Chen Ling, Junji Jiang, Junxiang Wang, My T. Thai, Renhao Xue, James Song, Meikang Qiu, and Liang Zhao. Deep graph representation learning and optimization for influence maximization. In *Proc. of the 40th International Conference on Machine Learning (ICML 2023)*, volume 202 of *Proceedings of Machine Learning Research*, pages 21350–21361, 2023. URL: https://proceedings.mlr.press/v202/ling23b.html.

15 Takanori Maehara, Hirofumi Suzuki, and Masakazu Ishihata. Exact computation of influence spread by binary decision diagrams. In *Proc. of the 26th International Conference on World Wide Web (WWW 2017)*, pages 947–956, 2017. doi:10.1145/3038912.3052567.

16 Kengo Nakamura, Takeru Inoue, Masaaki Nishino, and Norihito Yasuda. Efficient network reliability evaluation for client-server model. In *Proc. of the 2021 IEEE Global Communications Conference (GLOBECOM 2021)*, pages 1–6, 2021. doi:10.1109/GLOBECOM46510.2021.9685283.

17 Kengo Nakamura, Takeru Inoue, Masaaki Nishino, Norihito Yasuda, and Shin-ichi Minato. A fast and exact evaluation algorithm for the expected number of connected nodes: an enhanced network reliability measure. In *Proc. of the 2023 IEEE International Conference on Computer Communications (INFOCOM 2023)*, pages 1–10, 2023. doi:10.1109/INFOCOM53939.2023.10228897.

18 Naoto Ohsaka, Takuya Akiba, Yuichi Yoshida, and Ken-ichi Kawarabayashi. Fast and accurate influence maximization on large networks with pruned monte-carlo simulations. In *Proc. of the 28th AAAI Conference on Artificial Intelligence (AAAI 2014)*, pages 138–144, 2014. doi:10.1609/aaai.v28i1.8726.

19 Shashank Sheshar Singh, Divya Srivastva, Madhushi Verma, and Jagendra Singh. Influence maximization frameworks, performance, challenges and directions on social network: A theoretical study. *Journal of King Saud University - Computer and Information Sciences*, 34(9):7570–7603, 2022. doi:10.1016/j.jksuci.2021.08.009.

20 Hirofumi Suzuki, Masakazu Ishihata, and Shin-ichi Minato. Exact computation of strongly connected reliability by binary decision diagrams. In *Proc. of the 12th International Conference on Combinatorial Optimization and Applications (COCOA 2018)*, pages 281–295, 2018. doi:10.1007/978-3-030-04651-4_19.

21 Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972. doi:10.1137/0201010.

22 Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979. doi:10.1137/0208032.

23 Peng Wu and Li Pan. Scalable influence blocking maximization in social networks under competitive independent cascade models. *Computer Networks*, 123:38–50, 2017. doi:10.1016/j.comnet.2017.05.004.

24 Mao Ye, Xingjie Liu, and Wang-Chien Lee. Exploring social influence for recommendation: a generative model approach. In *Proc. of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2012)*, pages 671–680, 2012. doi:10.1145/2348283.2348373.