Scalable Influence Estimation in Continuous-Time Diffusion Networks

Nan Du, Le Song and Hongyuan Zha
Georgia Institute of Technology
dunan@gatech.edu
{lsong, zha}@cc.gatech.edu

Manuel Gomez Rodriguez
Max Planck Institute for Cybernetics
m.gomez.rodriguez@gmail.com

Abstract

If a piece of information is released from a media site, can it spread, in 1 month, to a million web pages? This influence estimation problem is very challenging since both the time-sensitive nature of the problem and the issue of scalability need to be addressed simultaneously. In this paper, we propose a randomized algorithm for influence estimation in continuous-time diffusion networks. Our algorithm can estimate the influence of every node in a network with $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges to an accuracy of ϵ using $n = O(1/\epsilon^2)$ randomizations and up to logarithmic factors $O(n|\mathcal{E}| + n|\mathcal{V}|)$ computations. When used as a subroutine in a greedy influence maximization algorithm, our proposed algorithm is guaranteed to find a set of C nodes with an influence of at least $(1 - 1/e)\,\mathrm{OPT} - 2C\epsilon$, where OPT is the optimal value. Experiments on both synthetic and real-world data show that the proposed algorithm can easily scale up to networks of millions of nodes while significantly improves over previous state-of-the-arts in terms of the accuracy of the estimated influence and the quality of the selected nodes in maximizing the influence.

1 Introduction

Motivated by applications viral marketing [1], researchers have been studying the influence maximization problem: find a set of nodes whose initial adoptions of certain idea or product can trigger, *in a time window*, the largest expected number of follow-ups. For this purpose, it is essential to accurately and efficiently estimate the number of follow-ups of an arbitrary set of source nodes within the given time window.

This is a challenging problem since we need to first accurately model the timing information in cascade data, and second design an efficient algorithm which can scale up to networks with millions of nodes. Most previous work in the literature tackled the influence estimation and maximization problems for infinite time window [2, 3, 4, 5, 1, 6, 7]. However, in most cases, influence must be estimated or maximized up to a given time, *i.e.*, a finite time window must be considered [8]. For example, a marketer would like to have her advertisement viewed by a million people in one month, rather than in one hundred years. Such time-sensitive requirement renders those algorithms which only consider static information, such as network topologies, inappropriate for this context.

A sequence of recent work has argued that modeling cascade data and information diffusion using *continuous-time* diffusion networks can provide significantly more accurate models than discrete-time models [9, 10, 11, 12, 13, 14, 15]. There is a twofold rationale behind this modeling choice. First, since follow-ups occur asynchronously, continuous variables seem more appropriate to represent them. Artificially discretizing the time axis into bins introduces additional tuning parameters, like the bin size, which are not easy to choose optimally. Second, discrete time models can only model transmission times which obey an exponential density, and hence can be too restricted to capture the rich temporal dynamics in the data. Extensive experimental comparisons on both synthetic and real world data showed that continuous-time models yield significant improvement in settings

such as recovering hidden diffusion network structures from cascade data [9, 11] and predicting the timings of future events [10, 15].

However, estimating and maximizing influence based on continuous-time diffusion models also entail significant challenges. First, the influence estimation problem in this setting is a difficult graphical model inference problem, *i.e.*, computing the marginal density of continuous variables in loopy graphical models. The exact answer can be computed only for very special cases. For example, Gomez-Rodriguez et al. [8] have shown that the problem can be solved exactly when the transmission functions are exponential densities, by using continuous time Markov processes theory. However, the computational complexity of such approach, in general, scales exponentially with the size and density of the network and, moreover, extending the approach to deal with arbitrary transmission functions would require additional nontrivial approximations which would increase even more the computational complexity. Second, it is unclear how to scale up influence estimation and maximization algorithms based on continuous-time diffusion models to millions of nodes. Especially in the maximization case, even a naive sampling algorithm for approximate inference is not scalable: n sampling rounds need to be carried out for each node to estimate the influence, which results in an overall $O(n|\mathcal{V}||\mathcal{E}|)$ algorithm. Thus, our goal is to design a scalable algorithm which can perform influence estimation and maximization in this regime of networks with millions of nodes.

In particular, we propose CONTINEST (**Con**tinous-**T**ime **In**fluence **Est**imation), a scalable randomized algorithm for influence estimation in a continuous-time diffusion network with heterogeneous edge transmission functions. The key idea of the algorithm is to view the problem from the perspective of graphical model inference, and reduces the problem to a neighborhood estimation problem in graphs. Our algorithm can estimate the influence of every node in a network with $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges to an accuracy of ϵ using $n = O(1/\epsilon^2)$ randomizations and up to logarithmic factors $O(n|\mathcal{E}| + n|\mathcal{V}|)$ computations. When used as a subroutine in a greedy influence maximization algorithm, our proposed algorithm is guaranteed to find a set of nodes with an influence of at least $(1 - 1/e)\,\mathrm{OPT} - 2C\epsilon$, where OPT is the optimal value. Finally, we validate CONTINEST on both influence estimation and maximization problems over large synthetic and real world datasets. In terms of influence estimation, CONTINEST is much closer to the true influence and much faster than other state-of-the-art methods. With respect to the influence maximization, CONTINEST allows us to find a set of sources with greater influence than other state-of-the-art methods.

2 Continuous-Time Diffusion Networks

First, we revisit the continuous-time generative model for cascade data in social networks introduced in [11]. The model associates each edge $j \to i$ with a transmission function, $f_{ji}(\tau_{ji})$, a density over time, in contrast to previous discrete-time models which associate each edge with a fixed infection probability [1]. Moreover, it also differs from discrete-time models in the sense that events in a cascade are not generated iteratively in rounds, but event timings are sampled directly from the transmission function in the continuous-time model. More specifically,

Continuous-Time Independent Cascade Model. Given a *directed* contact network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we use a continuous-time independent cascade model for modeling a diffusion process [11]. The process begins with a set of infected source nodes, $\mathcal{A}$, initially adopting certain *contagion* (idea, meme or product) at time zero. The contagion is transmitted from the sources along their out-going edges to their direct neighbors. Each transmission through an edge entails a *random* transmission time, τ, drawn from a density over time, $f_{ji}(\tau)$. We assume transmission times are independent and possibly distributed differently across edges. Then, the infected neighbors transmit the contagion to their respective neighbors, and the process continues. We assume that an infected node remains infected for the entire diffusion process. Thus, if a node i is infected by multiple neighbors, only the neighbor that first infects node i will be the *true parent*. As a result, although the contact network can be an arbitrary directed network, each cascade (a vector of event timing information from the spread of a contagion) induces a Directed Acyclic Graph (DAG).

Heterogeneous Transmission Functions. Formally, the transmission function $f_{ji}(t_i|t_j)$ for directed edge $j \to i$ is the conditional density of node i getting infected at time t_i given that node j was infected at time t_j. We assume it is shift invariant: $f_{ji}(t_i|t_j) = f_{ji}(\tau_{ji})$, where $\tau_{ji} := t_i - t_j$, and nonnegative: $f_{ji}(\tau_{ji}) = 0$ if $\tau_{ji} < 0$. Both parametric transmission functions, such as the exponential and Rayleigh function [11, 16], and nonparametric function [9] can be used and estimated from cascade data (see Appendix A for more details).

Shortest-Path property. The independent cascade model has a useful property we will use later: given a sample of transmission times of all edges, the time t_i taken to infect a node i is the length of the shortest path in $\mathcal{G}$ from the sources to node i, where the edge weights correspond to the associated transmission times.

3 Graphical Model Perspectives for Continuous-Time Diffusion Networks

The continuous-time independent cascade model is essentially a directed graphical model for a set of *dependent* random variables, the infection times t_i of the nodes, where the conditional independence structure is supported on the contact network $\mathcal{G}$ (see Appendix B for more details). More formally, the joint density of $\{t_i\}_{i\in\mathcal{V}}$ can be expressed as

$$p\left(\{t_i\}_{i\in\mathcal{V}}\right) = \prod\nolimits_{i\in\mathcal{V}} p\left(t_i|\{t_j\}_{j\in\pi_i}\right), \tag{1}$$

where π_i denotes the set of parents of node i in a cascade-induced DAG, and $p(t_i|\{t_j\}_{j\in\pi_i})$ is the conditional density of infection t_i at node i given the infection times of its parents.

Instead of directly modeling the infection times t_i, we can focus on the set of mutually *independent* random transmission times $\tau_{ji} = t_i - t_j$. Interestingly, by switching from a node-centric view to an edge-centric view, we obtain a fully factorized joint density of the set of transmission times

$$p\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) = \prod\nolimits_{(j,i)\in\mathcal{E}} f_{ji}(\tau_{ji}), \tag{2}$$

Based on the Shortest-Path property of the independent cascade model, each variable t_i can be viewed as a transformation from the collection of variables $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$.

More specifically, let $\mathcal{Q}_i$ be the collection of directed paths in $\mathcal{G}$ from the source nodes to node i, where each path $q \in \mathcal{Q}_i$ contains a sequence of directed edges (j, l). Assuming all source nodes are infected at zero time, then we obtain variable t_i via

$$t_i = g_i\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) := \min_{q\in\mathcal{Q}_i} \sum\nolimits_{(j,l)\in q} \tau_{jl}, \tag{3}$$

where the transformation $g_i(\cdot)$ is the value of the shortest-path minimization. As a special case, we can now compute the probability of node i infected before T using a set of independent variables:

$$\Pr\{t_i \le T\} = \Pr\left\{g_i\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) \le T\right\}. \tag{4}$$

The significance of the relation is that it allows us to transform a problem involving a sequence of dependent variables $\{t_i\}_{i\in\mathcal{V}}$ to one with independent variables $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$. Furthermore, the two perspectives are connected via the shortest path algorithm in weighted directed graph, a standard well-studied operation in graph analysis.

4 Influence Estimation Problem in Continuous-Time Diffusion Networks

Intuitively, given a time window, the wider the spread of infection, the more influential the set of sources. We adopt the definition of influence as the average number of infected nodes given a set of source nodes and a time window, as in previous work [8]. More formally, consider a set of C source nodes $\mathcal{A} \subseteq \mathcal{V}$ which gets infected at time zero, then, given a time window T, a node i is infected in the time window if $t_i \le T$. The expected number of infected nodes (or the influence) given the set of transmission functions $\{f_{ji}\}_{(j,i)\in\mathcal{E}}$ can be computed as

$$\sigma(\mathcal{A}, T) = \mathbb{E}\left[\sum\nolimits_{i\in\mathcal{V}} \mathbb{I}\{t_i \le T\}\right] = \sum\nolimits_{i\in\mathcal{V}} \mathbb{E}\left[\mathbb{I}\{t_i \le T\}\right] = \sum\nolimits_{i\in\mathcal{V}} \Pr\{t_i \le T\}, \tag{5}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function and the expectation is taken over the the set of *dependent* variables $\{t_i\}_{i\in\mathcal{V}}$.

Essentially, the influence estimation problem in Eq. (5) is an inference problem for graphical models, where the probability of event $t_i \le T$ given sources in $\mathcal{A}$ can be obtained by summing out the possible configuration of other variables $\{t_j\}_{j\neq i}$. That is

$$\Pr\{t_i \le T\} = \int_0^\infty \cdots \int_{t_i=0}^{T} \cdots \int_0^\infty \left(\prod\nolimits_{j\in\mathcal{V}} p\left(t_j|\{t_l\}_{l\in\pi_j}\right)\right)\left(\prod\nolimits_{j\in\mathcal{V}} dt_j\right) \tag{6}$$

, which is, in general, a very challenging problem. First, the corresponding directed graphical models can contain nodes with high in-degree and high out-degree. For example, in Twitter, a user can follow dozens of other users, and another user can have hundreds of "followees". The tree-width

corresponding to this directed graphical model can be very high, and we need to perform integration for functions involving many continuous variables. Second, the integral in general can not be evaluated analytically for heterogeneous transmission functions, which means that we need to resort to numerical integration by discretizing the domain $[0, \infty)$. If we use N level of discretization for each variable, we would need to enumerate $O(N^{|\pi_i|})$ entries, exponential in the number of parents.

Only in very special cases, can one derive the closed-form equation for computing $\Pr\{t_i \leq T\}$ [8]. However, without further heuristic approximation, the computational complexity of the algorithm is exponential in the size and density of the network. The intrinsic complexity of the problem entails the utilization of approximation algorithms, such as mean field algorithms or message passing algorithms.We will design an efficient randomized (or sampling) algorithm in the next section.

5 Efficient Influence Estimation in Continuous-Time Diffusion Networks

Our first key observation is that we can transform the influence estimation problem in Eq. (5) into a problem with *independent* variables. Using relation in Eq. (4), we have

$$\sigma(\mathcal{A}, T) = \sum\nolimits_{i \in \mathcal{V}} \Pr\left\{g_i\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) \leq T\right\} = \mathbb{E}\left[\sum\nolimits_{i \in \mathcal{V}} \mathbb{I}\left\{g_i\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) \leq T\right\}\right], \quad (7)$$

where the expectation is with respect to the set of independent variables $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$. This equivalent formulation suggests a naive sampling (NS) algorithm for approximating $\sigma(\mathcal{A}, T)$: draw n samples of $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$, run a shortest path algorithm for each sample, and finally average the results (see Appendix C for more details). However, this naive sampling approach has a computational complexity of $O(nC|\mathcal{V}||\mathcal{E}| + nC|\mathcal{V}|^2 \log |\mathcal{V}|)$ due to the repeated calling of the shortest path algorithm. This is quadratic to the network size, and hence not scalable to millions of nodes.

Our second key observation is that for each sample $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$, we are only interested in the neighborhood size of the source nodes, *i.e.*, the summation $\sum_{i\in\mathcal{V}} \mathbb{I}\{\cdot\}$ in Eq. (7), rather than in the individual shortest paths. Fortunately, the neighborhood size estimation problem has been studied in the theoretical computer science literature. Here, we adapt a very efficient randomized algorithm by Cohen [17] to our influence estimation problem. This randomized algorithm has a computational complexity of $O(|\mathcal{E}| \log |\mathcal{V}| + |\mathcal{V}| \log |\mathcal{V}|)$ and it estimates the neighborhood sizes for *all* possible single source node locations. Since it needs to run once for each sample of $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$, we obtain an overall influence estimation algorithm with $O(n|\mathcal{E}| \log |\mathcal{V}| + n|\mathcal{V}| \log^2 |\mathcal{V}|)$ computation, nearly linear in network size. Next we will revisit Cohen's algorithm for neighborhood estimation.

5.1 Randomized Algorithm for Single-Source Neighborhood Size Estimation

Given a fixed set of edge transmission times $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$ and a source node s, infected at time 0, the neighborhood $\mathcal{N}(s, T)$ of a source node s given a time window T is the set of nodes within distance T from s, *i.e.*,

$$\mathcal{N}(s, T) = \left\{i \;\middle|\; g_i\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) \leq T,\ i \in \mathcal{V}\right\}. \quad (8)$$

Instead of estimating $\mathcal{N}(s, T)$ directly, the algorithm will assign an exponentially distributed random label r_i to each network node i. Then, it makes use of the fact that the minimum of a set of exponential random variables $\{r_i\}_{i\in\mathcal{N}(s,T)}$ will also be a exponential random variable, but with its parameter equals to the number of variables. That is if each $r_i \sim \exp(-r_i)$, then the smallest label within distance T from source s, $r_* := \min_{i\in\mathcal{N}(s,T)} r_i$, will distribute as $r_* \sim \exp\{-|\mathcal{N}(s, T)|r_*\}$. Suppose we randomize over the labeling m times, and obtain m such least labels, $\{r_*^u\}_{u=1}^m$. Then the neighborhood size can be estimated as

$$|\mathcal{N}(s, T)| \approx \frac{m-1}{\sum_{u=1}^m r_*^u}. \quad (9)$$

which is shown to be an unbiased estimator of $|\mathcal{N}(s, T)|$ [17]. This is an interesting relation since it allows us to transform the counting problem in (8) to a problem of finding the minimum random label r_*. The key question is whether we can compute the least label r_* efficiently, given random labels $\{r_i\}_{i\in\mathcal{V}}$ and any source node s.

Cohen [17] designed a modified Dijkstra's algorithm (Algorithm 1) to construct a data structure $r_*(s)$, called least label list, for each node s to support such query. Essentially, the algorithm starts with the node i with the smallest label r_i, and then it traverses in breadth-first search fashion along the reverse direction of the graph edges to find all reachable nodes. For each reachable node s, the distance d_* between i and s, and r_i are added to the end of $r_*(s)$. Then the algorithm moves to the

node i' with the second smallest label $r_{i'}$, and similarly find all reachable nodes. For each reachable node s, the algorithm will compare the current distance d_* between i' and s with the last recorded distance in $r_*(s)$. If the current distance is smaller, then the current d_* and $r_{i'}$ are added to the end of $r_*(s)$. Then the algorithm move to the node with the third smallest label and so on. The algorithm is summarized in Algorithm 1 in Appendix D.

Algorithm 1 returns a list $r_*(s)$ per node $s \in \mathcal{V}$, which contains information about distance to the smallest reachable labels from s. In particular, each list contains pairs of distance and random labels, (d, r), and these pairs are ordered as

$$\infty > d_{(1)} > d_{(2)} > \ldots > d_{(|r_*(s)|)} = 0 \tag{10}$$

$$r_{(1)} < r_{(2)} < \ldots < r_{(|r_*(s)|)}, \tag{11}$$

where $\{\cdot\}_{(l)}$ denotes the l-th element in the list. (see Appendix D for an example). If we want to query the smallest reachable random label r_* for a given source s and a time T, we only need to perform a binary search on the list for node s:

$$r_* = r_{(l)}, \text{ where } d_{(l-1)} > T \geq d_{(l)}. \tag{12}$$

Finally, to estimate $|\mathcal{N}(s, T)|$, we generate m *i.i.d.* collections of random labels, run Algorithm 1 on each collection, and obtain m values $\{r_*^u\}_{u=1}^m$, which we use on Eq. (9) to estimate $|\mathcal{N}(i, T)|$.

The computational complexity of Algorithm 1 is $O(|\mathcal{E}| \log |\mathcal{V}| + |\mathcal{V}| \log^2 |\mathcal{V}|)$, with expected size of each $r_*(s)$ being $O(\log |\mathcal{V}|)$. Then the expected time for querying r_* is $O(\log \log |\mathcal{V}|)$ using binary search. Since we need to generate m set of random labels and run Algorithm 1 m times, the overall computational complexity for estimating the single-source neighborhood size for all $s \in \mathcal{V}$ is $O(m|\mathcal{E}| \log |\mathcal{V}| + m|\mathcal{V}| \log^2 |\mathcal{V}| + m|\mathcal{V}| \log \log |\mathcal{V}|)$. For large scale network, and when $m \ll \min\{|\mathcal{V}|, |\mathcal{E}|\}$, this randomized algorithm can be much more efficient than approaches based on directly calculating the shortest paths.

5.2 Constructing Estimation for Multiple-Source Neighborhood Size

When we have a set of sources, $\mathcal{A}$, its neighborhood is the union of the neighborhoods of its constituent sources

$$\mathcal{N}(\mathcal{A}, T) = \bigcup\nolimits_{i \in \mathcal{A}} \mathcal{N}(i, T). \tag{13}$$

This is true because each source independently infects its downstream nodes. Furthermore, to calculate the least label list r_* corresponding to $\mathcal{N}(\mathcal{A}, T)$, we can simply reuse the least label list $r_*(i)$ of each individual source $i \in \mathcal{A}$. More formally,

$$r_* = \min_{i \in \mathcal{A}} \ \min_{j \in \mathcal{N}(i,T)} r_j, \tag{14}$$

where the inner minimization can be carried out by querying $r_*(i)$. Similarly, after we obtain m samples of r_*, we can estimate $|\mathcal{N}(\mathcal{A}, T)|$ using Eq. (9). Importantly, very little additional work is needed when we want to calculate r_* for a set of sources $\mathcal{A}$, and we can reuse work done for a single source. This is very different from a naive sampling approach where the sampling process needs to be done completely anew if we increase the source set. In contrast, using the randomized algorithm, only an additional constant-time minimization over $|\mathcal{A}|$ numbers is needed.

5.3 Overall Algorithm

So far, we have achieved efficient neighborhood size estimation of $|\mathcal{N}(\mathcal{A}, T)|$ with respect to a given set of transmission times $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$. Next, we will estimate the influence by averaging over multiple sets of samples for $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$. More specifically, the relation from (7)

$$\sigma(\mathcal{A}, T) = \mathbb{E}_{\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}} \left[|\mathcal{N}(\mathcal{A}, T)|\right] = \mathbb{E}_{\{\tau_{ji}\}} \mathbb{E}_{\{r^1, \ldots, r^m\} | \{\tau_{ji}\}} \left[\frac{m-1}{\sum_{u=1}^m r_*^u} \right], \tag{15}$$

suggests the following overall algorithm

Continuous-Time Influence Estimation (CONTINEST):

1. Sample n sets of random transmission times $\{\tau_{ij}^l\}_{(j,i)\in\mathcal{E}} \sim \prod_{(j,i)\in\mathcal{E}} f_{ji}(\tau_{ji})$
2. Given a set of $\{\tau_{ij}^l\}_{(j,i)\in\mathcal{E}}$, sample m sets of random labels $\{r_i^u\}_{i\in\mathcal{V}} \sim \prod_{i\in\mathcal{V}} \exp(-r_i)$
3. Estimate $\sigma(\mathcal{A}, T)$ by sample averages $\sigma(\mathcal{A}, T) \approx \frac{1}{n} \sum_{l=1}^n \left((m-1) / \sum_{u_l=1}^m r_*^{u_l} \right)$

Importantly, the number of random labels, m, does not need to be very large. Since the estimator for $|\mathcal{N}(A,T)|$ is unbiased [17], essentially the outer-loop of averaging over n samples of random transmission times further reduces the variance of the estimator in a rate of $O(1/n)$. In practice, we can use a very small m (*e.g.*, 5 or 10) and still achieve good results, which is also confirmed by our later experiments. More formally, we have the following guarantee (see Appendix E for proof)

Theorem 1 *Draw the following number of samples for the set of random transmission times*

$$n \geqslant \frac{C\Lambda}{\epsilon^2} \log\left(\frac{2|\mathcal{V}|}{\delta}\right) \tag{16}$$

where $\Lambda := \max_{\mathcal{A}:|\mathcal{A}|\leq C} 2\sigma(\mathcal{A},T)/(m-2) + 2Var(|\mathcal{N}(\mathcal{A},T)|) + 2a\epsilon/3$ *and* $|\mathcal{N}(\mathcal{A},T)| \leq a$*, and for each set of random transmission times, draw* m *set of random labels. Then* $|\widehat{\sigma}(\mathcal{A},T) - \sigma(\mathcal{A},T)| \leqslant \epsilon$ *uniformly for all* $\mathcal{A}$ *with* $|\mathcal{A}| \leqslant C$*, with probability at least* $1-\delta$*.*

The theorem indicates that the minimum number of samples, n, needed to achieve certain accuracy is related to the actual size of the influence $\sigma(\mathcal{A},T)$, and the variance of the neighborhood size $|\mathcal{N}(\mathcal{A},T)|$ over the random draw of samples. The number of random labels, m, drawn in the inner loop of the algorithm will monotonically decrease the dependency of n on $\sigma(\mathcal{A},T)$. It suffices to draw a small number of random labels, as long as the value of $\sigma(\mathcal{A},T)/(m-2)$ matches that of $Var(|\mathcal{N}(\mathcal{A},T)|)$. Another implication is that influence at larger time window T is harder to estimate, since $\sigma(\mathcal{A},T)$ will generally be larger and hence require more random labels.

6 Influence Maximization

Once we know how to estimate the influence $\sigma(\mathcal{A},T)$ for any $\mathcal{A} \subseteq \mathcal{V}$ and time window T efficiently, we can use them in finding the optimal set of C source nodes $\mathcal{A}^* \subseteq \mathcal{V}$ such that the expected number of infected nodes in $\mathcal{G}$ is maximized at T. That is, we seek to solve,

$$\mathcal{A}^* = \text{argmax}_{|\mathcal{A}|\leqslant C}\ \sigma(\mathcal{A},T), \tag{17}$$

where set $\mathcal{A}$ is the variable. The above optimization problem is NP-hard in general. By construction, $\sigma(\mathcal{A},T)$ is a non-negative, monotonic nondecreasing function in the set of source nodes, and it can be shown that $\sigma(\mathcal{A},T)$ satisfies a diminishing returns property called submodularity [8].

A well-known approximation algorithm to maximize monotonic submodular functions is the *greedy algorithm*. It adds nodes to the source node set $\mathcal{A}$ sequentially. In step k, it adds the node i which maximizes the *marginal gain* $\sigma(\mathcal{A}_{k-1} \cup \{i\}; T) - \sigma(\mathcal{A}_{k-1}; T)$. The greedy algorithm finds a source node set which achieves at least a constant fraction $(1-1/e)$ of the optimal [18]. Moreover, lazy evaluation [6] can be employed to reduce the required number of *marginal gains* per iteration. By using our influence estimation algorithm in each iteration of the greedy algorithm, we gain the following additional benefits:

First, at each iteration k, we do not need to rerun the full influence estimation algorithm (section 5.2). We just need to store the least label list $r_*(i)$ for each node $i \in \mathcal{V}$ computed for a single source, which requires expected storage size of $O(|\mathcal{V}| \log |\mathcal{V}|)$ overall.

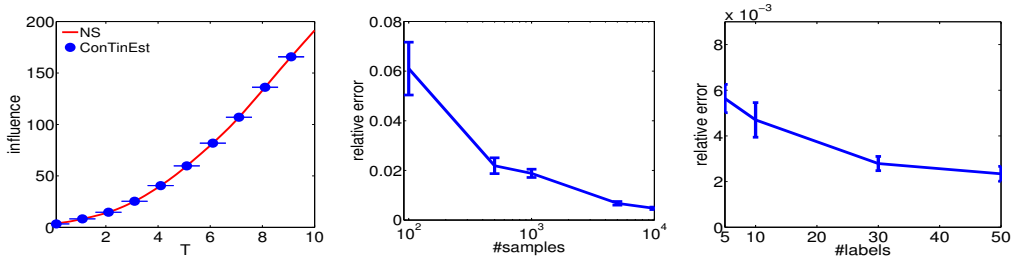
Second, our influence estimation algorithm can be easily parallelized. Its two nested sampling loops can be parallelized in a straightforward way since the variables are independent of each other. However, in practice, we use a small number of random labels, and $m \ll n$. Thus we only need to parallelize the sampling for the set of random transmission times $\{\tau_{ji}\}$. The storage of the least element lists can also be distributed.

However, by using our randomized algorithm for influence estimation, we also introduce a sampling error to the greedy algorithm due to the approximation of the influence $\sigma(\mathcal{A},T)$. Fortunately, the greedy algorithm is tolerant to such sampling noise, and a well-known result provides a guarantee for this case (following an argument in [19, Th. 7.9]):

Theorem 2 *Suppose the influence* $\sigma(\mathcal{A},T)$ *for all* $\mathcal{A}$ *with* $|\mathcal{A}| \leq C$ *are estimated uniformly with error* ϵ *and confidence* $1-\delta$*, the greedy algorithm returns a set of sources* $\widehat{\mathcal{A}}$ *such that* $\sigma(\widehat{\mathcal{A}},T) \geq (1-1/e)OPT - 2C\epsilon$ *with probability at least* $1-\delta$*.*

7 Experiments

We first evaluate the accuracy of the estimated influence given by CONTINEST on several synthetic networks. Then, by incorporating CONTINEST into the greedy algorithm described above, we investigate the overall performance of influence maximization on several synthetic and real networks.



(a) Influence vs. time (b) Error vs. #samples (c) Error vs. #labels

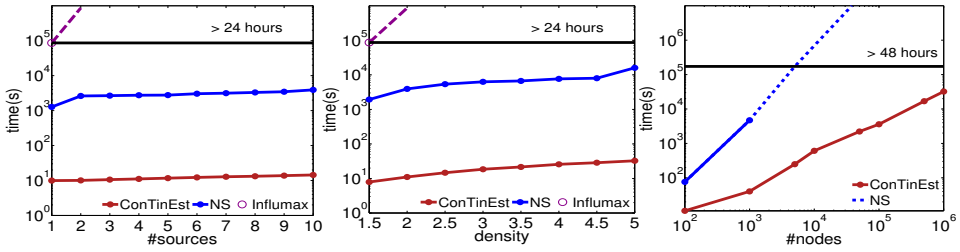
Figure 1: For core-periphery networks with 1,024 nodes and 2,048 edges, (a) estimated influence for increasing time window T, and (b) fixing $T = 10$, relative error for increasing number of samples with 5 random labels, and (c) for increasing number of random labels with 10,000 random samples.

We show that our approach significantly outperforms the state-of-the-art methods in terms of both speed and solution quality.

Synthetic network generation. We generate three types of Kronecker networks [20]: (*i*) core-periphery networks (parameter matrix: [0.9 0.5; 0.5 0.3]), which mimic the information diffusion traces in real world networks [21], (*ii*) random networks ([0.5 0.5; 0.5 0.5]), typically used in physics and graph theory [22] and (*iii*) hierarchical networks ([0.9 0.1; 0.1 0.9]) [11]. Next, we assign a pairwise transmission function for every directed edge in each type of network and set its parameters at random. In our experiments, we use the Weibull distribution [16], $f(t; \alpha, \beta) = \frac{\beta}{\alpha} \left(\frac{t}{\alpha}\right)^{\beta-1} e^{-(t/\alpha)^{\beta}}, t \geqslant 0$, where $\alpha > 0$ is a scale parameter and $\beta > 0$ is a shape parameter. The Weibull distribution (Wbl) has often been used to model lifetime events in survival analysis, providing more flexibility than an exponential distribution [16]. We choose α and β from 0 to 10 uniformly at random for each edge in order to have heterogeneous temporal dynamics. Finally, for each type of Kronecker network, we generate 10 sample networks, each of which has different α and β chosen for every edge.

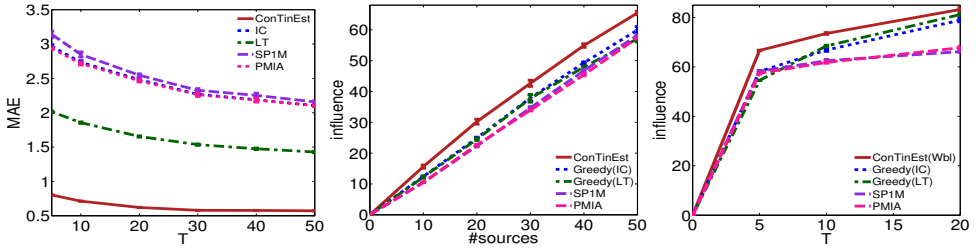
Accuracy of the estimated influence. To the best of our knowledge, there is no analytical solution to the influence estimation given Weibull transmission function. Therefore, we compare CONTINEST with Naive Sampling (NS) approach (see Appendix C) by considering the highest degree node in a network as the source, and draw 1,000,000 samples for NS to obtain near ground truth. Figures 1(a) compares CONTINEST with the ground truth provided by NS at different time window T, from 0.1 to 10 in corre-periphery networks. For CONTINEST, we generate up to 10,000 random samples (or set of random waiting times), and 5 random labels in the inner loop. In all three networks, estimation provided by CONTINEST fits accurately the ground truth, and the relative error decreases quickly as we increase the number of samples and labels (Figures 1(b) and 1(c)). For 10,000 random samples with 5 random labels, the relative error is smaller than 0.01. (see Appendix F for additional results on the random and hierarchal networks)

Scalability. We compare CONTINEST to previous state-of-the-art INFLUMAX [8] and the Naive Sampling (NS) method in terms of run time for the continuous-time influence estimation and maximization. For CONTINEST, we draw 10,000 samples in the outer loop, each having 5 random labels in the inner loop. For NS, we also draw 10,000 samples. The first two experiments are carried out in a single 2.4GHz processor. First, we compare the performance for increasing number of selected sources (from 1 to 10) by fixing the core-periphery networks to 128 node network and 320 edges and time window to 10 (Figure 2(a)). When the number of selected sources is 1, different algorithms essentially spend time estimating the influence for each node. CONTINEST outperforms other methods by order of magnitude and for the number of sources larger than 1, it can efficiently reuse computations for estimating influence for individual nodes. Dashed lines mean that a method did not finish in 24 hours, and the estimated run time is plotted. Next, we compare the run time for selecting 10 sources on core-periphery networks of 128 nodes with increasing densities (or the number of edges) (Figure 2(a)). Again, INFLUMAX and NS are order of magnitude slower due to their respective exponential and quadratic computational complexity in network density. In contrast, the run time of CONTINEST only increases slightly with the increasing density since its computational complexity is linear in the number of edges. (see Appendix F for additional results on the random and hierarchal networks)



(a) Run time vs. # sources (b) Run time vs. network density (c) Run time vs. #nodes

Figure 2: For core-periphery networks, (a) runtime for selecting increasing number of sources in networks of 128 nodes and 320 edges with time window 10. (b) runtime for selecting 10 sources in networks of 128 nodes with increasing density with time window 10. (c) runtime for selecting 10 nodes in networks with nodes size increasing from 100 to 1,000,000 and with density 1.5 and time window 10.



(a) Influence estimation error (b) Influence vs. #sources (c) Influence vs. time

Figure 3: In MemeTracker dataset, (a) comparison of the accuracy of the estimated influence in terms of mean absolute error, (b) comparison of the influence of the selected nodes by fixing the observation window $T = 5$ and varying the number sources, and (c) comparison of the influence of the selected nodes by by fixing the number of sources to 50 and varying the time window.

Finally, we evaluate the speed on large core-periphery networks, ranging from 100 to 1,000,000 nodes with density 1.5 in Figure 2(c). We report the parallel run time only for CONTINEST and NS (both are implemented by MPI running on 192 cores of 2.4Ghz) since INFLUMAX is not scalable. In contrast to NS, the performance of CONTINEST increases linearly with the network size and can easily scale up to one million nodes.

Real-world data. We first quantify how well each method can estimate the true influence in a real-world dataset. Then, we further evaluate the solution quality of the selected sources for the influence maximization problem. We use MemeTracker dataset [23], in which we have extracted 10,967 hyperlink cascades among the top 600 media sites, each of which is a collection of time-stamps which record the creation of posts and articles referring to each other about similar pieces of information. We randomly split all the cascades into a 80% training set and a 20% test set and repeat the random splitting for five times in total. Then, on each training set, we infer the parameter for continuous-time model by using NETRATE [11] and exponential transmission functions. For discrete-time model, we learn the infection probabilities using [24] for IC, SP1M and PMIA. Similarly, for LT, we follow the methodology by [1], as in our previous experiments. Then, we evaluate the influence on the test set as follows: given node u, let $\mathcal{C}(u)$ be the set of all cascades in which u was the source node. Then based on the cascades in $\mathcal{C}(u)$, the total number of distinct nodes infected before T quantifies the real influence of node u up to time T. In Figure 3(a), we report the Mean Absolute Error (MAE) between the real and the estimated influence for different observation window T and methods (which use the estimated influence for future maximization). Clearly, CONTINEST estimates the real influence of each source node significantly better than competitive models. Since the estimation of the real influence of each source node is the foundation for the influence maximization problem, we also expect the greedy method with CONTINEST to find a set of sources with higher influence. Figures 3(b) and 3(c) confirm this intuition. We evaluate the influence of the selected nodes in the same spirit as influence estimation: the true influence is calculated as the total number of distinct nodes infected before T based on $\mathcal{C}(u)$ of the selected nodes. The selected sources given by CONTINEST achieve the best performance as we vary the number of selected sources and the observation time window.

8 Conclusions

In this paper, we propose a randomized nearly linear time algorithm, CONTINEST, for influence estimation in continuous-time diffusion networks. Experiments on both synthetic and real-world data show that the proposed algorithm can easily scale up to networks of millions of nodes while significantly improves over previous state-of-the-arts in terms of the accuracy of the estimated influence and the quality of the selected nodes in maximizing the influence. There are also many venues for future work. It will be interesting to apply the current algorithm to other task such as influence minimization and influence manipulation, and design scalable randomized algorithms for continuous-time models other than the independent cascade model.

References

[1] David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.

[2] Shishir Bharathi, David Kempe, and Mahyar Salek. Competitive influence maximization in social networks. In *WINE*, pages 306–311, 2007.

[3] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *KDD*, pages 199–208, 2009.

[4] Wei Chen, Yifei Yuan, and Li Zhang. Scalable influence maximization in social networks under the linear threshold model. In *ICDM*, pages 88–97, 2010.

[5] Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. A data-based approach to social influence maximization. *Proc. VLDB Endow.*, 5, 2011.

[6] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne M. VanBriesen, and Natalie S. Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429, 2007.

[7] Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*, pages 61–70, 2002.

[8] Manuel Gomez-Rodriguez and Bernhard Schölkopf. Influence maximization in continuous time diffusion networks. In *ICML '12*, 2012.

[9] Nan Du, Le Song, Alexander J. Smola, and Ming Yuan. Learning networks of heterogeneous influence. In *NIPS*, 2012.

[10] Nan Du, Le Song, Hyenkyun Woo, and Hongyuan Zha. Uncover topic-sensitive information diffusion networks. In *AISTATS*, 2013.

[11] Manuel Gomez-Rodriguez, David Balduzzi, and Bernhard Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *ICML*, pages 561–568, 2011.

[12] M. Gomez-Rodriguez, J. Leskovec, and B. Schölkopf. Structure and Dynamics of Information Pathways in On-line Media. In *WSDM*, 2013.

[13] K. Zhou, L. Song, and H. Zha. Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes. In *Artificial Intelligence and Statistics (AISTATS)*, 2013.

[14] K. Zhou, H. Zha, and L. Song. Learning triggering kernels for multi-dimensional hawkes processes. In *International Conference on Machine Learning(ICML)*, 2013.

[15] M. Gomez-Rodriguez, J. Leskovec, and B Schölkopf. Modeling information propagation with survival theory. In *ICML*, 2013.

[16] Jerald F. Lawless. *Statistical Models and Methods for Lifetime Data*. Wiley-Interscience, 2002.

[17] Edith Cohen. Size-estimation framework with applications to transitive closure and reachability. *Journal of Computer and System Sciences*, 55(3):441–453, 1997.

[18] GL Nemhauser, LA Wolsey, and ML Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1), 1978.

[19] Andreas Krause. *Ph.D. Thesis*. CMU, 2008.

[20] Jure Leskovec, Deepayan Chakrabarti, Jon M. Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *JMLR*, 11, 2010.

[21] Manuel Gomez-Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. In *KDD*, 2010.

[22] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

[23] Jure Leskovec, Lars Backstrom, and Jon M. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD*, 2009.

[24] Praneeth Netrapalli and Sujay Sanghavi. Learning the graph of epidemic cascades. In *SIGMETRICS/PERFORMANCE*, pages 211–222. ACM, 2012.

[25] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD '10*, pages 1029–1038, 2010.

A Heterogeneous Transmission Functions

We denote the waiting time distribution, or transmission function, along a directed edge of $\mathcal{G}$ as $f_{ji}(t_i|t_j)$. Formally, the transmission function $f_{ji}(t_i|t_j)$ for directed edge $j \to i$ is the conditional density of node i getting infected at time t_i given that node j was infected at time t_j. We assume it is shift invariant, *i.e.*, $f_{ji}(t_i|t_j) = f_{ji}(t_i - t_j) = f_{ji}(\tau_{ji})$, where $\tau_{ji} := t_i - t_j$, and it takes positive values when $\tau_{ji} \geq 0$, and the value of zero otherwise.

In most previous work, simple parametric transmission functions such as the exponential distribution $\alpha_{ji} \exp(-\alpha_{ji}\tau_{ji})$, and the Rayleigh distribution $\alpha_{ji}\tau \exp(-\alpha_{ji}\tau_{ji}^2/2)$ have been used [16, 11]. However, in many real world scenarios, information transmission between pairs of nodes can be heterogeneous and the waiting times can obey distributions that dramatically differ from these simple models. For instance, in viral marketing, active consumers could update their status instantly, while an inactive user may just log in and respond once a day. As a result, the transmission function between an active user and his friends can be quite different from that between an inactive user and his friends. As an attempt to model these complex scenarios, nonparametric transmission functions have been recently considered [9]. In such approach, the relationship between the survival function, the conditional intensity function or hazard, and the transmission function is exploited. In particular, the survival function is defined as $S_{ji}(\tau_{ji}) := 1 - \int_0^{\tau_{ji}} f_{ji}(\tau')d\tau'$ and the hazard function is defined as $h_{ji}(\tau_{ji}) := f_{ji}(\tau_{ji})/S_{ji}(\tau_{ji})$. Then, it is a well-known result in survival theory that $S_{ji}(\tau_{ji}) = \exp\left(-\int_0^{\tau_{ji}} h_{ji}(\tau')d\tau'\right)$ and $f_{ji}(\tau_{ji}) = h_{ji}(\tau_{ji})S_{ji}(\tau_{ji})$. The advantage of using the conditional intensity function is that we do not need to explicitly enforce "the integral equals 1" constraint for the conditional density f_{ji}. Instead, we just need to ensure $h_{ji} \geq 0$. This facilitates nonparametric modeling of the transmission function. For instance, we can define the conditional intensity function as a positive combination of n positive kernel functions k,

$$h_{ji}(\tau) = \sum\nolimits_{l=1}^{n} \alpha_l k(\tau_l, \tau), \text{ if } \tau > 0, \text{ and 0 otherwise.}$$

A common choice of the kernel function is the Gaussian RBF kernel $k(\tau', \tau) = \exp(-\left\|\tau - \tau'\right\|^2 / 2s^2)$. Nonparametric transmission functions significantly improve modeling of real world diffusion, as is shown in [9].

B A Graphical Model Perspective

Now, we look at the independent cascade model from the perspective of graphical models, where the collection of random variables includes the infection times t_i of the nodes. Although the original contact graph $\mathcal{G}$ can contain directed loops, each diffusion process (or a cascade) induces a directed acyclic graph (DAG). For those cascades consistent with a particular DAG, we can model the joint density of t_i using a directed graphical model:

$$p\left(\{t_i\}_{i\in\mathcal{V}}\right) = \prod\nolimits_{i\in\mathcal{V}} p\left(t_i|\{t_j\}_{j\in\pi_i}\right), \tag{18}$$

where each π_i denotes the collection of parents of node i in the induced DAG, and each term $p(t_i|\{t_j\}_{j\in\pi_i})$ corresponds to a conditional density of t_j given the infection times of the parents of node i. This is true because given the infection times of node i's parents, t_i is independent of other infection times, satisfying the local Markov property of a directed graphical model. We note that the independent cascade model only specifies explicitly the pairwise transmission functions for each directed edge, but does not directly define the conditional density $p(t_i|\{t_j\}_{j\in\pi_i})$.

However, these conditional densities can be derived from the pairwise transmission functions based on the Independent-Infection property [11]:

$$p\left(t_i|\{t_j\}_{j\in\pi_i}\right) = \sum\nolimits_{j\in\pi_i} h_{ji}(t_i|t_j) \prod\nolimits_{l\in\pi_i} S(t_i|t_l), \tag{19}$$

which is the sum of the likelihoods that node i is infected by each parent node j. More precisely, each term in the summation can be interpreted as the instantaneous risk of node i being infected at t_i by node j given that it has survived the infection of all parent nodes until time t_i.

Perhaps surprisingly, the factorization in Eq. (18) is the same factorization that can be used for an arbitrary induced DAG consistent with the contact network $\mathcal{G}$. In this case, we only need to replace the definition of π_i (the parent of node i in the DAG) to the set of neighbors of node i with an edge

pointing to node i in $\mathcal{G}$. This is not immediately obvious from Eq. (18), since the contact network $\mathcal{G}$ can contain directed loops which may be in conflict with the conditional independence semantics of directed graphical models. The reason it is possible to do so is as follows: Any fixed set of infection times, $t_1, \ldots, t_d$, induces an ordering of the infection times. If $t_i \leq t_j$ for an edge $j \to i$ in $\mathcal{G}$, $h_{ji}(t_i|t_j) = 0$, and the corresponding term in Eq. (19) is zeroed out, making the conditional density consistent with the semantics of directed graphical models.

Based on the joint density of the infection times in Eq. (18), we can perform various inference and learning tasks. For instance, previous work has used Eq. (18) for learning the parameters of the independent cascade model [9, 11, 12]. However, this may not be the most convenient form for addressing other inference problems, including the influence estimation problem in the next section. To this end, we propose an alternative view.

Instead of directly modeling the infection times t_i, we can focus on the collection of mutually independent random transmission times $\tau_{ji} = t_i - t_j$. In this case, the joint density of the collection of transmission times τ_{ji} is fully factorized

$$p\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) = \prod\nolimits_{(j,i)\in\mathcal{E}} f_{ji}(\tau_{ji}),$$

where $\mathcal{E}$ denotes the set of edges in the contact network $\mathcal{G}$ — switching from the earlier node-centric view to the now edge-centric view. Based on the Shortest-Path property of the independent cascade model, variable t_i can be viewed as a transformation from the collection of variables $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$. More specifically, let $\mathcal{Q}_i$ be the collection of directed paths in $\mathcal{G}$ from the source nodes to node i, where each path $q \in \mathcal{Q}_i$ contains a sequence of directed edges (j, l), and assuming all source nodes are infected at zero time, then we obtain variable t_i via

$$t_i = g_i\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) := \min_{q\in\mathcal{Q}_i} \sum\nolimits_{(j,l)\in q} \tau_{jl}, \tag{20}$$

where $g_i(\cdot)$ is the transformation.

Importantly, we can now compute the probability of infection of node i at t_i using the set of variables $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$:

$$\Pr\{t_i \leq T\} = \Pr\left\{g_i\left(\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}\right) \leq T\right\}. \tag{21}$$

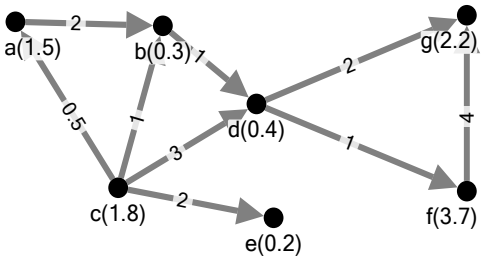
The significance of the relation is that it allows us to transform a problem involving a sequence of dependent variables $\{t_i\}_{i\in\mathcal{V}}$ to one with independent variables $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$. Furthermore, the two problems are connected via the shortest path algorithm in weighted directed graph, a standard well studied operation in graph analysis.

C Naive Sampling Algorithm

The graphical model perspective described in Section 3 and Appendix B suggests a naive sampling (NS) algorithm for approximating $\sigma(\mathcal{A}, T)$:

1. Draw n samples, $\left\{\left\{\tau_{ji}^l\right\}_{(j,i)\in\mathcal{E}}\right\}_{l=1}^n$, *i.i.d.* from the waiting time product distribution $\prod_{(j,i)\in\mathcal{E}} f_{ji}(\tau_{ji})$;
2. For each sample $\left\{\tau_{ji}^l\right\}_{(j,i)\in\mathcal{E}}$ and for each node i, find the shortest path from source nodes to node i; count the number of nodes with $g_i\left(\left\{\tau_{ji}^l\right\}_{(j,i)\in\mathcal{E}}\right) \leq T$;
3. Average the counts across n samples.

Although the naive sampling algorithm can handle arbitrary transmission function, it is not scalable to networks with millions of nodes. We need to compute the shortest path for each node and each sample, which results in a computational complexity of $O(n|\mathcal{E}| + n|\mathcal{V}|\log|\mathcal{V}|)$ for a single source node. The problem is even more pressing in the influence maximization problem, where we need to estimate the influence of source nodes at different location and with increasing number of source nodes. To do this, the algorithm needs to be repeated, adding a multiplicative factor of $C|\mathcal{V}|$ to the computational complexity (C is the number of nodes to select). Then, the algorithm becomes quadratic in the network size. When the network size is in the order of thousands and millions, typical in modern social network analysis, the naive sampling algorithm become prohibitively ex-



- Node labeling :
  $e(0.2) < b(0.3) < d(0.4) < a(1.5) < c(1.8) < g(2.2) < f(3.7)$
- Neighborhoods:
  $\mathcal{N}(c, 2) = \{a, b, c, e\}$; $\mathcal{N}(c, 3) = \{a, b, c, d, e, f\}$;
- Least-label list:
  $r_*(c) : (2, 0.2), (1, 0.3), (0.5, 1.5), (0, 1.8)$
- Query: $r_*(c, 0.8) = r(a) = 1.5$

Figure 4: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, edge weights $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$, and node labeling $\{r_i\}_{i\in\mathcal{V}}$ with the associated output from Algorithm 1.

pensive. Additionally, we may need to draw thousands of samples (n is large), further making the algorithm impractical for large scale problems.

D Least Label List

The notation "argsort$((r_1, \ldots, r_{|\mathcal{V}|}), \text{ascend})$" in line 2 of Algorithm 1 means that we sort the collection of random labels in ascending order and return the argument of the sort as an ordered list.

Algorithm 1: Least Label List

Input: a reversed directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with edge weights $\{\tau_{ji}\}_{(j,i)\in\mathcal{E}}$, a node labeling $\{r_i\}_{i\in\mathcal{V}}$
Output: A list $r_*(s)$ for each $s \in \mathcal{V}$
for *each* $s \in \mathcal{V}$ **do** $d_s \leftarrow \infty, r_*(s) \leftarrow \emptyset$
for i *in argsort*$((r_1, \ldots, r_{|\mathcal{V}|}), \textit{ascend})$ **do**
  empty heap H $\leftarrow \emptyset$;
  set all nodes except i as unvisited;
  push $(0, i)$ into heap H;
  while H $\neq \emptyset$ **do**
    pop (d_*, s) with the minimum d_* from H;
    add (d_*, r_i) to the end of list $r_*(s)$;
    $d_s \leftarrow d^*$;
    for *each unvisited in-coming neighbor* j *of* s **do**
      set j as visited;
      if (d, j) *in heap* H **then**
        Pop (d, j) from heap H;
        Push $(\min\{d, d_* + \tau_{js}\}, j)$ into heap H;
      else if $d_* + \tau_{js} < d_j$ **then**
        Push $(d_* + \tau_{js}, j)$ into heap H;

Figure 4 shows an example of the Least-Label-List. The nodes from a to g are assigned to exponentially distributed labels with mean one shown in each parentheses. Given a query distance 0.8 for node c, we can binary-search its Least-label-list $r_*(c)$ to find that node a belongs to this range with the smallest label $r(a) = 1.5$.

E Theorem 1

Theorem 1 *Sample the following number of sets of random transmission times*

$$n \geqslant \frac{C\Lambda}{\epsilon^2} \log\left(\frac{2|\mathcal{V}|}{\delta}\right)$$

where $\Lambda := \max_{\mathcal{A}:|\mathcal{A}|\leq C} 2\sigma(\mathcal{A},T)/(m-2) + 2Var(S_\tau) + 2a\epsilon/3$*, and for each set of random transmission times, sample* m *set of random labels, we can guarantee that*

$$|\widehat{\sigma}(\mathcal{A},T) - \sigma(\mathcal{A},T)| \leqslant \epsilon$$

simultaneously for all $\mathcal{A}$ *with* $|\mathcal{A}| \leqslant C$*, with probability at least* $1-\delta$*.*

Proof Let $S_\tau := |\mathcal{N}(\mathcal{A},T)|$ for a fixed set of $\{\tau_{ji}\}$ and then $\sigma(\mathcal{A},T) = \mathbb{E}_\tau[S_\tau]$. The randomized algorithm with m randomizations produces an unbiased estimator $\widehat{S}_\tau = (m-1)/(\sum_{u=1}^m r_*^u)$ for S_τ, *i.e.*, $\mathbb{E}_{r|\tau}[\widehat{S}_\tau] = S_\tau$, with variance $\mathbb{E}_{r|\tau}[(\widehat{S}_\tau - S_\tau)^2] = S_\tau/(m-2)$.

Then $\widehat{S}_\tau$ is also an unbiased estimator for $\sigma(\mathcal{A},T)$, since $\mathbb{E}_{\tau,r}[\widehat{S}_\tau] = \mathbb{E}_\tau\mathbb{E}_{r|\tau}[\widehat{S}_\tau] = \mathbb{E}_\tau[S_\tau] = \sigma(\mathcal{A},T)$. Its variance is

$$\begin{aligned}
Var(\widehat{S}_\tau) &:= \mathbb{E}_{\tau,r}[(\widehat{S}_\tau - \sigma(\mathcal{A},T))^2] = \mathbb{E}_{\tau,r}[(\widehat{S}_\tau - S_\tau + S_\tau - \sigma(\mathcal{A},T))^2] \\
&= \mathbb{E}_{\tau,r}[(\widehat{S}_\tau - S_\tau)^2] + 2\,\mathbb{E}_{\tau,r}[(\widehat{S}_\tau - S_\tau)(S_\tau - \sigma(\mathcal{A},T))] + \mathbb{E}_{\tau,r}[(S_\tau - \sigma(\mathcal{A},T))^2] \\
&= \mathbb{E}_\tau[S_\tau/(m-2)] + 0 + Var(S_\tau) \\
&= \sigma(\mathcal{A},T)/(m-2) + Var(S_\tau)
\end{aligned}$$

Then using Bernstein's inequality, we have, for our final estimator $\widehat{\sigma}(\mathcal{A},T) = \frac{1}{n}\sum_{l=1}^n \widehat{S}_{\tau^l}$, that

$$\Pr\left\{|\widehat{\sigma}(\mathcal{A},T) - \sigma(\mathcal{A},T)| \geqslant \epsilon\right\} \leqslant 2\exp\left(-\frac{n\epsilon^2}{2Var(\widehat{S}_\tau) + 2a\epsilon/3}\right) \tag{22}$$

where $\widehat{S}_\tau < a \leqslant |\mathcal{V}|$.

Setting the right hand side of relation (22) to δ, we have that, with probability $1-\delta$, sampling the following number set of random transmission times

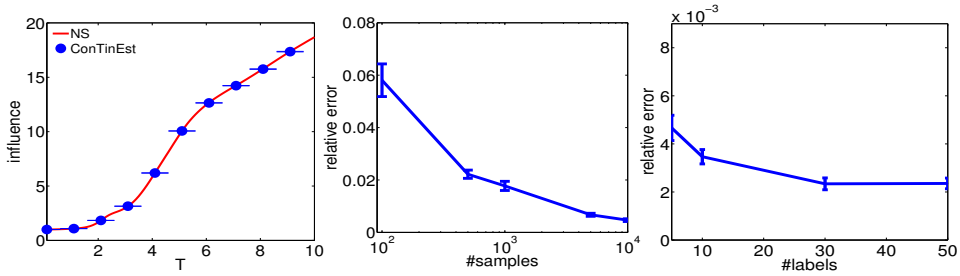
$$n \geqslant \frac{2Var(\widehat{S}_\tau) + 2a\epsilon/3}{\epsilon^2}\log\left(\frac{2}{\delta}\right) = \frac{2\sigma(\mathcal{A},T)/(m-2) + 2Var(S_\tau) + 2a\epsilon/3}{\epsilon^2}\log\left(\frac{2}{\delta}\right)$$

we can guarantee that our estimator to have error $|\widehat{\sigma}(\mathcal{A},T) - \sigma(\mathcal{A},T)| \leqslant \epsilon$.

If we want to insure that $|\widehat{\sigma}(\mathcal{A},T) - \sigma(\mathcal{A},T)| \leqslant \epsilon$ simultaneously hold for all $\mathcal{A}$ such that $|\mathcal{A}| \leqslant C \ll |\mathcal{V}|$, we can first use union bound with relation (22). In this case, we have that, with probability $1-\delta$, sampling the following number set of random transmission times

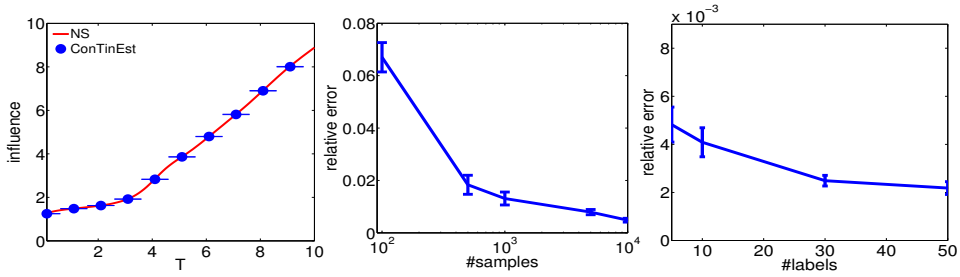
$$n \geqslant \frac{C\Lambda}{\epsilon^2}\log\left(\frac{2|\mathcal{V}|}{\delta}\right)$$

we can guarantee that our estimator to have error $|\widehat{\sigma}(\mathcal{A},T) - \sigma(\mathcal{A},T)| \leqslant \epsilon$ for all $\mathcal{A}$ with $|\mathcal{A}| \leqslant C$. Note that we have define the constant $\Lambda := \max_{\mathcal{A}:|\mathcal{A}|\leq C} 2\sigma(\mathcal{A},T)/(m-2) + 2Var(S_\tau) + 2a\epsilon/3$. ■



(a) Influence vs. time (b) Error vs. #samples (c) Error vs. #labels

Figure 5: On the **random** kronecker networks with 1,024 nodes and 2,048 edges, panels show (a) the estimated influence with increasing time window T; (b) the average relative error for different number of samples, each of which has 5 random labels for every node; and (c) the average relative error for varying number of random labels assigned to every node in each of 10,000 samples. For both (b) and (c), we set $T = 10$.



(a) Influence vs. time (b) Error vs. #samples (c) Error vs. #labels

Figure 6: On the **hierarchical** kronecker networks with 1,024 nodes and 2,048 edges, panels show (a) the estimated influence with increasing time window T; (b) the average relative error for different number of samples, each of which has 5 random labels for every node; and (c) the average relative error for varying number of random labels assigned to every node in each of 10,000 samples. For both (b) and (c), we set $T = 10$.

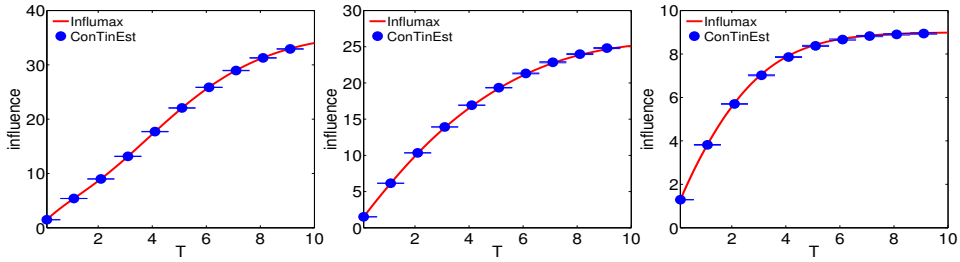
F Additional Experimental Results

In this section, we report additional experimental results on accuracy of influence estimation, continuous-time influence maximization and scalability for the synthetic networks.

F.1 Accuracy of Influence Estimation

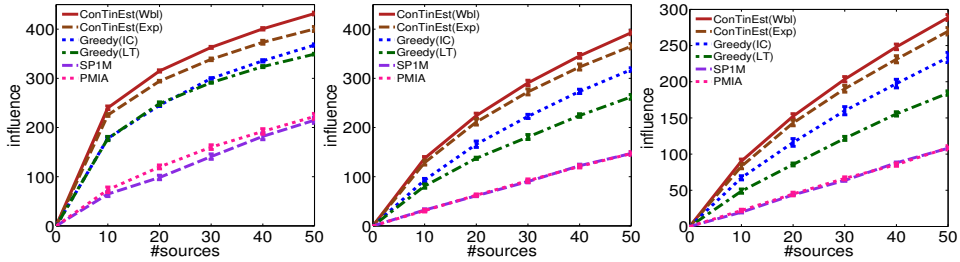
Figure 5 evaluates the estimated scope of influence for different time windows and the relative errors with respective to different number of random samples and labels on the random kronecker networks with 1,024 nodes and 2,048 edges. Figure 6 further reports similar results on the hierarchical kronecker networks. In all cases, the errors decrease dramatically as we draw more samples and labels.

In addition, because INFLUMAX can produce exact closed form influence on sparse small networks with exponential transmission functions, we compare CONTINEST with INFLUMAX in Figure 7, where we chose the highest degree node in the network as the source. We have drawn 10,000 random samples, each of which has 5 random labels for each node. CONTINEST outputs values of influence which are very close to the exact values given by INFLUMAX, with relative error less than 0.01 in all three types of networks.



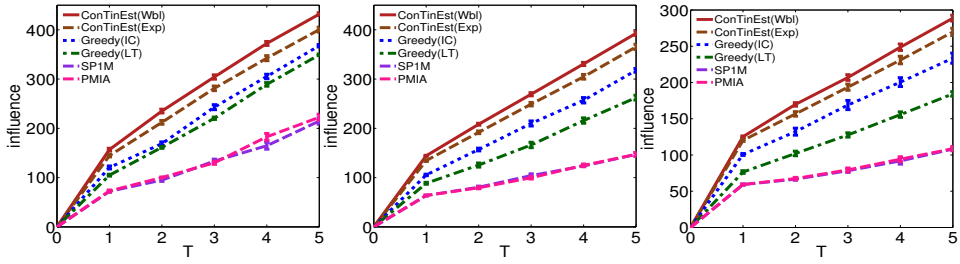
(a) Core-periphery (b) Random (c) Hierarchical

Figure 7: Infected neighborhood size over three different types of networks with the exponential transmission function associated with each edge. Each type of network consists of 128 nodes and 141 edges. For panels (d-i), we set the observation window $T = 10$.



(a) Core-periphery (b) Random (c) Hierarchical

Figure 8: Panels present the influence against the number of sources by $T = 5$ on the networks having 1,024 nodes and 2,048 edges with heterogeneous Weibull transmission functions.

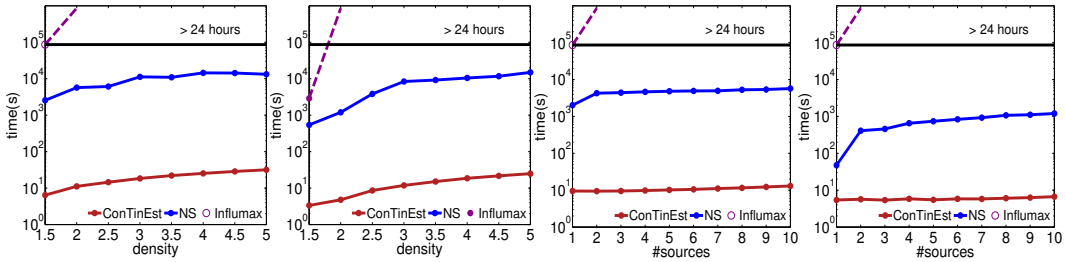


(a) Core-periphery (b) Random (c) Hierarchal

Figure 9: Panels present the influence against the time window T using 50 sources on the networks having 1,024 nodes and 2,048 edges with heterogeneous Weibull transmission functions.

F.2 Continuous-time Influence Maximization

We compare CONTINEST to other influence maximization methods based on discrete-time diffusion models: traditional greedy [1], with discrete-time Linear Threshold Model (LT) and Independent Cascade Model (IC) diffusion models, and the heuristic methods SP1M [3] and PMIA [25]. For INFLUMAX, since it only supports exponential pairwise transmission functions, we fit an exponential distribution per edge. Furthermore, INFLUMAX is not scalable; when the average network density of the synthetic networks is ~ 2.0, the run time for INFLUMAX is larger than 24 hours. Instead, we present the results of CONTINEST using fitted exponential distributions (Exp). For the discrete-time IC model, we learn the infection probability within time window T using Netrapalli's method [24]. The learned pairwise infection probabilities are also served for SP1M and PMIA, which essentially approximately calculate the influence based on the IC model. For the discrete-time LT model, we set the weight of each incoming edge to a node u to the inverse of its in-degree, as in previous work [1], and choose each node's threshold uniformly at random. Figure 8 compares the expected number of infected nodes against source set size for different methods. CONTINEST outperforms the rest, and the competitive advantage becomes more dramatic the larger the source set grows. Figure 9 shows



(a) Random (b) Hierarchal (c) Random (d) Hierarchal

Figure 10: Panels(a-b) show the running time against the network density by fixing the number of sources at 10 on the random and hierarchal kronecker network with 128 nodes. Panels(c-d) present the running time as we increase the number of selected source nodes on the networks with 128 nodes and 256 edges.

the expected number of infected nodes against the time window for 50 selected sources. Again, CONTINEST performs the best for all three types of networks.

F.3 Scalability

Figure 10 compares CONTINEST to INFLUMAX and the Naive Simulation (NS) method in terms of running time for the continuous-time influence maximization problem over the random and hierarchal kronecker type of networks, respectively, with different densities and sizes on a single 2.4Ghz CPU core. For CONTINEST, we have drawn 10,000 samples, each of which has 5 random labels assigned to each node. For NS, we follow the work [1] to run 10,000 Monte Carlo simulations. For running times longer than 24 hours, we use dashed line to qualitatively indicate the estimated performance based on the time complexity of each method.