Inferring Pairwise Influence from Encrypted Communication

Brian Thompson
U.S. Army Research Lab
2800 Powder Mill Road
Adelphi, MD 20783
bthompso8784@gmail.com

Hasan Cam
U.S. Army Research Lab
2800 Powder Mill Road
Adelphi, MD 20783
hasan.cam.civ@mail.mil

***Abstract*—Inferring influence in networks from observed communication activity is an important task in contexts such as surveillance, marketing, and cybersecurity. Most existing approaches rely on content or meta-data indicating related activity, rendering those approaches ineffective when such information is unavailable, for example due to encrypted communication. In contrast, we present an efficient algorithm to infer influence between entities that relies only on the times of their individual activity, paying particular attention to the computational challenges posed by large, high-volume networks. We provide theoretical bounds on the recall and runtime of our algorithm relative to characteristics of network structure and dynamics, along with experiments to support our theoretical analysis and validate the effectiveness of our approach.**

I. Introduction

A. Motivation

Many real-world systems—including communication, information, and tactical networks—consist of a set of entities, such as people or devices, and a sequence of events, each occurring at a particular time and involving one or more entities. For example, events could be email or SMS messages, radio broadcasts, device failures, or criminal or terrorist acts. Understanding the influences that entities exert on one another can help to expose organizational hierarchies or identify paths of information flow. When such influences are not known a priori, one may hope to infer their existence or strength by analyzing the past events in which the entities have been involved.

Many approaches for analyzing event-driven networks project continuous-time data onto a discrete-time model, for example by binning data over time windows, which introduces information loss and makes subsequent analysis dependent on the time scale used. We use a continuous-time stochastic model, which gives greater flexibility in modeling temporal processes and avoids the drawbacks of discretization.

Much previous work studying influence in networks focuses on aggregate behavior or the impact each entity has on the network as a whole. We model the pairwise relationships between entities explicitly, allowing us to capture differences in relationship dynamics across entity pairs, which better reflects the heterogeneity observed in real-world networks.

Some existing methods to infer influence in networks leverage content (e.g. co-occurring phrases, links, or hashtags) or meta-data indicating causal relationships between events (e.g. forwarded emails or retweets) [7], [9]. In application domains where such information is non-existent or unavailable, however, those methods cannot be applied. For example, malicious activities may be observed without indicating how they are connected. Device failures in a cyber network are often logged, but the paths of exploited vulnerabilities leading to them may not be known. Monitoring a communication network can reveal when messages are sent, but message contents may be encrypted.

Given a set of entities in a network and their times of activity, our goal is to determine the influences driving their behavior. We use the Expectation-Maximization (EM) algorithm to quantify pairwise influences between entities and other aspects of network dynamics, leveraging only the temporal information inherent in the data rather than relying on additional content or meta-data. Previous work in this line of research has so far focused on analyzing the behavior of individuals or very small networks due to the computational challenges involved. We build on their work, developing an approach that is computationally feasible for large, high-volume networks. In addition, we provide theoretical bounds on our algorithm's recall and runtime relative to properties of the network structure and dynamics, along with experiments to support our theoretical analysis and validate the effectiveness of our approach. Our analysis can help practitioners gauge the computational resources needed and the quality of results to expect from using our approach to analyze their network data.

B. Related Work

Gomez-Rodriguez et al. [3] present a model for information diffusion in networks, along with an algorithm to infer the most likely network structure based on a set of known cascades (sequences of events by which information spreads through a network), learning both the existence of edges and their transmission rates. They demonstrate that their method is able to accurately learn the network structure given a sufficiently large collection of known cascades. In contrast, our approach does not require a compiled list of labeled cascades.

In the early 1970s, Hawkes [5] introduced self- and mutually exciting temporal point processes to model scenarios in which the rate of activity of a process changes dynamically in

response to past activity of itself or others. Because such models do not express causal relationships between events explicitly, instead attributing the occurrence of an event to the confluence of all previous activity, they avoid the need for content or meta-data indicating related events. In many real-world contexts, however, causal relationships between events are believed to exist, and it may be useful to model or infer them even if they are not explicit in the data.

Veen and Schoenberg [8] model the occurrence of earthquakes along a fault line as a self-exciting process, treating causal relationships between earthquakes and their aftershocks as missing data and estimating the model parameters using the EM algorithm. Halpin and De Boeck [4] model dyadic interactions as pairs of self- and mutually exciting processes and use EM to infer the influence that two people have on one another. Fox et al. [2] consider influences among twenty-two people in an email network, where each person sends and receives emails at roughly the same rate. We broaden the scope to consider networks where each entity observes the activities of many others and the goal is to infer which of them exert a strong influence on the entity's behavior.

C. Contributions and Outline

The main contributions of this work are:

- an efficient algorithm to infer pairwise influence between entities in a network from encrypted communication data
- theoretical bounds on the recall and runtime of our algorithm dependent on network characteristics
- experimental results to support our theoretical analysis and validate the effectiveness of our approach

II. Methods

A. Model

We model network activity as being driven by a system of inter-dependent stochastic point processes expressing pairwise influences among a set of *entities* $\mathcal{U}$. As illustrated in Figure 1(a), each entity u performs some actions independent of others' activities according to a Poisson process with inter-event time distribution G_u with mean g_u, and also reacts to actions performed by a subset of the other entities after a lag time sampled from distribution L_u with mean l_u.

If entity u reacts to the actions of entity u', we say that u' *influences* u. The influential pairs are captured by the *reaction graph* R, shown in Figure 1(b), which contains the edge (u', u) if and only if u' influences u.

We represent each entity's actions as a sequence of instantaneous *events*, shown in Figure 1(c), where event $\varepsilon = (u, t)$ corresponds to an action taken by entity $u \in \mathcal{U}$ at time $t \in \mathbb{R}^+$. We use $\mathcal{E}$ to denote the set of all events. If event ε occurs in reaction to event ε', we say that ε' *triggers* ε.

B. Problem Statement

Network entities may communicate with one another via encrypted messages broadcast and received from secure devices. The sender and time of each message can be observed and recorded, but the underlying network structure and dynamics–including the intended recipients of the messages–are unknown. Our goal is to infer the network structure from the observed data, indicating the pairwise influences that entities exert on one another by triggering reactive behavior.

We frame our problem as a link prediction task: Let $\mathcal{U}$ be a set of entities whose behavior is described by the above model and whose pairwise influences are captured by the reaction graph R. Given a sequence of events $\mathcal{E}$ corresponding to their network activity, determine which ordered pairs of entities form an edge in R.

C. Approach

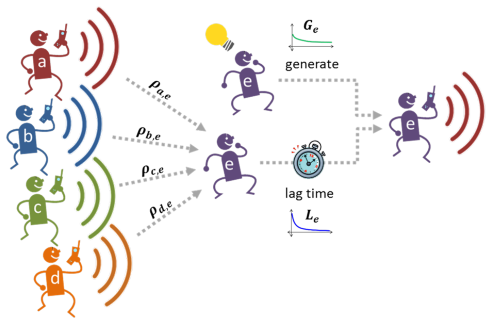
A natural approach is to set up a $\{0, 1\}$ integer program to determine the reaction graph with the greatest likelihood of having generated the observed data. However, integer programming is known to be NP-hard, making that approach impractical for many real-world contexts [6]. To address this computational problem, we propose a relaxation of the problem in which R is a weighted graph, with edge weights $\rho_{u',u}$ taking real values in $[0, 1]$. Intuitively, it may help to think of $\rho_{u',u}$ as the probability that u reacts to each action of u'. To infer the network structure, we then project the maximum likelihood solution onto the original space, predicting that $(u', u) \in R$ if and only if the maximum likelihood value $\widehat{\rho}_{u',u}$ is greater than a threshold γ.

Instead of computing the maximum likelihood values directly, we treat the triggers for the events as missing data and use Expectation-Maximization (EM) to infer the edge weights of the reaction graph and the generator and lag time distribution parameters for each entity—collectively referred to as the *model parameters*—and the event triggers simultaneously. The missing data provides additional structure to the problem, which helps guide the algorithm through the solution space in search of the global optimum. The intuition is as follows: Given estimates of the model parameters, we can compute the likelihood of any assignment of event triggers. Conversely, given a probability distribution over all possible trigger assignments, we can perform Maximum Likelihood Estimation (MLE) of the model parameters by marginalizing over the trigger assignments. The EM algorithm starts with an initial setting of the model parameters and then iterates between the two steps, each time improving the parameter estimates. We refer the reader to [1] for a more detailed explanation of the EM algorithm.

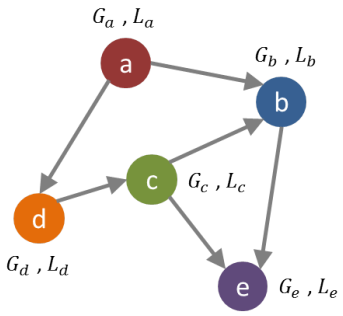
D. Algorithm

Let $\mathcal{E} = \{\varepsilon_i\}$ with $\varepsilon_i = (u_i, t_i)$ be the observed event data; let $\mathbf{Z} = \{Z_i\}$ be a set of random variables corresponding to the unknown event triggers, where $Z_i = j > 0$ denotes that event ε_i was triggered by event ε_j, and $Z_i = 0$ denotes that ε_i was generated independently; and let $\boldsymbol{\theta} = \{\rho_{u',u}\} \cup \{l_u\} \cup \{g_u\}$ be the model parameters.

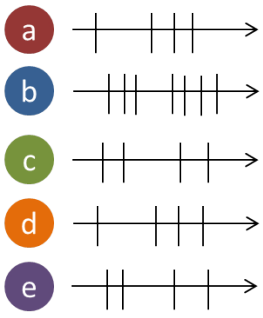
We use the Expectation-Maximization algorithm to infer the model parameters from the observed event data. The EM algorithm alternates between an Expectation step (E-step)



(a) Conceptual diagram of the influences on an entity e. Actions by e are either generated independently or in reaction to one of the other entities after some lag time.



(b) Visualization of the reaction graph R, along with the generator and lag time distributions for each entity.



(c) Event data, represented as a time sequence for each entity in the network.

Fig. 1: Three aspects of our model: (a) influences on an entity, (b) the reaction graph, and (c) event data.

and a Maximization step (M-step), iteratively improving the parameter estimates.

Consider the conditional probability density function over trigger assignments $\mathbf{Z}$ given event data $\mathcal{E}$ and model parameters $\boldsymbol{\theta}$, $f_{\mathbf{Z}|\mathcal{E};\boldsymbol{\theta}}$. Under our model, the triggers for different events are mutually independent given $\mathcal{E}$ and $\boldsymbol{\theta}$, so we have:

$$f_{\mathbf{Z}|\mathcal{E};\boldsymbol{\theta}} = \prod_{i=1}^{|\mathcal{E}|} f_{Z_i|\mathbf{Z}_{<i},\mathcal{E};\boldsymbol{\theta}} = \prod_{i=1}^{|\mathcal{E}|} f_{Z_i|\mathcal{E};\boldsymbol{\theta}}.$$

E-step: Given the observed event data $\mathcal{E}$ and initial parameter estimates $\widehat{\boldsymbol{\theta}}$, the likelihood of event ε_i having a particular trigger is computed as follows:

$$\begin{aligned}\mathcal{L}_{i,j} &= \mathcal{L}(Z_i = j) \\ &= f_{Z_i|\mathcal{E};\widehat{\boldsymbol{\theta}}}(j) \\ &= \begin{cases} 1/g_{u_i} & \text{if } j = 0 \\ 0 & \text{if } j > 0 \text{ and } t_j \geq t_i \\ \rho(u_j, u_i) \cdot PDF_{L_{u_i}}(t_i - t_j) & \text{otherwise} \end{cases}\end{aligned}$$

We normalize the likelihood values to get the weighted likelihoods, which naturally correspond to a probability distribution over all possible trigger assignments for ε_i:

$$\mathcal{W}_{i,j} = \frac{\mathcal{L}_{i,j}}{\sum_{k=0}^{|\mathcal{E}|} \mathcal{L}_{i,k}}.$$

M-step: Updated parameter estimates are then computed by marginalizing over the weighted likelihoods. Because of the independence property, the parameters for each entity can be computed separately. Let $\mathcal{E}_u = \{\varepsilon_i \in \mathcal{E} : u_i = u\}$. We have:

$$g_u = \frac{\text{duration of dataset}}{\sum_{i:\varepsilon_i \in \mathcal{E}_u} \mathcal{W}_{i,0}}$$

$$l_u = \frac{\sum_{i:\varepsilon_i \in \mathcal{E}_u} \sum_{j:\varepsilon_j \in \mathcal{E}_{u'}} \mathcal{W}_{i,j} \cdot (t_i - t_j)}{\sum_{i:\varepsilon_i \in \mathcal{E}_u} \sum_{j:\varepsilon_j \in \mathcal{E}_{u'}} \mathcal{W}_{i,j}}$$

$$\rho_{u',u} = \frac{\sum_{i:\varepsilon_i \in \mathcal{E}_u} \sum_{j:\varepsilon_j \in \mathcal{E}_{u'}} \mathcal{W}_{i,j}}{|\mathcal{E}_{u'}|}$$

E. Complexity

During the E-step, our algorithm computes the weighted likelihood of each possible trigger for each event, which takes $O(|\mathcal{E}|^2)$ time. In the M-step, these are aggregated to determine the updated parameter estimates for the generator distributions, lag time distributions, and edge weights, which takes $O(|\mathcal{E}|)$, $O(|\mathcal{E}|^2)$, and $O(|\mathcal{E}|^2)$ time, respectively. Therefore, the overall runtime of our algorithm is $O(|\mathcal{E}|^2)$.

To improve efficiency, we use a heuristic that reduces the number of pairs of events that get examined. Instead of considering every possible trigger for an event, we only look at the most recent preceding event for each other entity. That is, although our model permits an entity to react to an older event, we treat the probability of this occurring as negligible. This approximation makes sense when the lag time distributions are decreasing and not heavy-tailed, which is true in many real-world contexts. When those properties do not hold, other heuristics should be used. The runtime complexity using this heuristic is $O(|U| \cdot |\mathcal{E}|)$, which we validate experimentally in Section III-E.

III. Evaluation

A. Experimental Setup

As noted in Section II-D, the maximum likelihood parameters corresponding to the influences on each entity can be computed independently. Therefore, to simplify our analysis, we perform experiments on simulated event data over a synthetic network in which a single entity u observes the activity of others and reacts to a subset of them, according to the following parameters:

- n: number of entities observed by u
- k: number of entities to which u reacts
- g: mean of generator distribution (same for all nodes)
- l: mean of lag time distribution (for u)
- d: duration of time over which event data is collected

When not specified, we use the following default parameter values in our experiments: $n = 100$, $k = 5$, $g = 1000$, $l = 10$, and $d = 10000$. Conceptualizing the units to be on the scale of minutes, this roughly corresponds to each entity generating

1-2 new events per day over a period of one week, with a mean lag time of 10 minutes.

We model the generation of new activity as a Poisson process with generator distribution $G \sim Exp(1/g)$ and the lag times as being Exponentially distributed with $L \sim Exp(1/l)$. We analyze the effectiveness of our algorithm when the data does not fit these models in Section III-D. Future work could explore the use of other distribution models.

Our experimental methodology is as follows:

1) Choose values for the parameters n, k, g, l, and d.
2) Construct the reaction graph R as a random directed graph with n nodes of in-degree k.
3) Simulate network activity according to our model, with parameters g and l, for duration of time d, resulting in a sequence of events $\mathcal{E}$.
4) Use our algorithm to determine the maximum likelihood reaction graph $\widehat{R}$ from the event data $\mathcal{E}$.

We evaluate the results of our algorithm by comparing the inferred weighted graph $\widehat{R}$ with the constructed graph R, labeling each pair of entities (u', u) as one of the following:

TP (True Positive)	if $\widehat{\rho}_{u',u} > \gamma$ and $(u', u) \in R$
FP (False Positive)	if $\widehat{\rho}_{u',u} > \gamma$ and $(u', u) \notin R$
TN (True Negative)	if $\widehat{\rho}_{u',u} \leq \gamma$ and $(u', u) \notin R$
FN (False Negative)	if $\widehat{\rho}_{u',u} \leq \gamma$ and $(u', u) \in R$

where γ is the prediction threshold. When not specified, we use a default value of $\gamma = 0.5$ in our experiments.

Finally, we compute the *recall*, the fraction of links in R that are successfully identified, and *precision*, the fraction of predicted links that are indeed in R:

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

Ideally, an effective algorithm would achieve both high recall and high precision, correctly identifying exactly the entities that exert influence on u. We evaluate the effectiveness of our algorithm below.

B. Dependence on Network Characteristics

We derive a theoretical lower bound on the expected recall of our algorithm for the link prediction task described above after a single iteration of EM if the generator inter-event times and the lag times are exponentially distributed (proof omitted due to space constraints):

$$E[\text{Recall}] = E\left[\frac{TP}{TP + FN}\right] \geq \frac{g}{g + (2n + 1) \cdot l}.$$

To support our theoretical results, we perform a series of experiments to better understand how the recall and precision of our algorithm depend on characteristics of the network structure and dynamics. We run our algorithm on synthetic network data as described in Section III-A, varying each network parameter while holding the others constant. We plot the results in Figure 2.

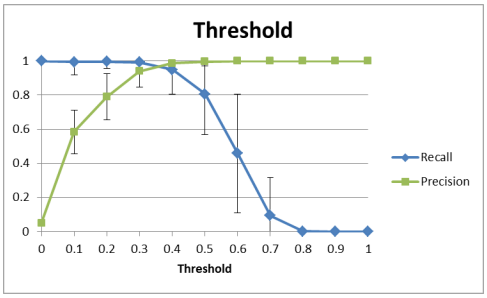


Fig. 3: Recall and precision across a range of prediction thresholds.

We find that recall is negatively correlated with n and l and positively correlated with g, which matches intuition as well as our theoretical analysis. Increasing n means more noise, corresponding to events that u observes but does not react to. Increasing l (the mean lag time) or decreasing g (which increases the rate at which each entity generates events) means that more activity will occur between when an event occurs and when u reacts, making it harder to correctly identify the event that triggered the reaction. The positive correlation with d is also intuitive: more data should improve results. The dependence on k is interesting, and is not reflected in our theoretical bounds. The initial decrease makes sense—larger k means greater competition in determining the correct trigger for an event among the influential entities—but the increase in recall as k approaches n is puzzling and deserves further investigation.

We note that precision is consistently high, indicating that the chance of incorrectly predicting a link (the false positive rate) is very low. It may therefore be possible to improve recall without significantly affecting precision by adjusting the prediction threshold γ. Figure 3 shows the results from running our algorithm on synthetic data as γ ranges from 0 to 1. We see that for the network parameters used in this experiment, balance is achieved at a threshold between 0.3 and 0.4, yielding approximately 97% recall and precision. We note that the shapes of the recall and precision curves depend on characteristics of the network and data and that the desired recall-precision tradeoff may be application-specific.

C. Convergence and Uniqueness

We examine the convergence rate of our algorithm, measured by the maximum value by which any $\widehat{\rho}$ parameter estimate changes during each iteration of EM. We find that with high probability, the maximum change falls below 0.01 after less than 20 iterations (see Figure 4).

Next we explore whether our algorithm converges to a unique solution or whether the results depend on the initial conditions used for EM. Figure 5 shows the results from running our algorithm multiple times on the same dataset with randomly chosen initial values. We see that our algorithm is indeed sensitive to the initial conditions. To address this, we

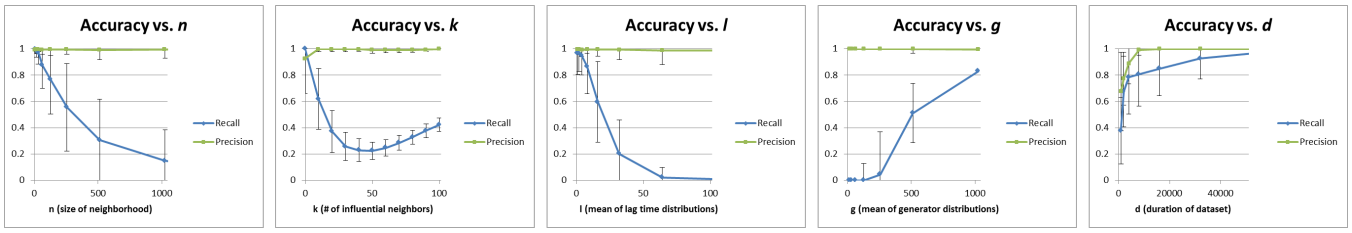


Fig. 2: Recall and precision demonstrating the sensitivity of our algorithm to various network parameters. Each plot varies one parameter, keeping the others fixed. Default values are: $n = 100$, $k = 5$, $g = 1000$, $l = 10$, and $d = 10000$.

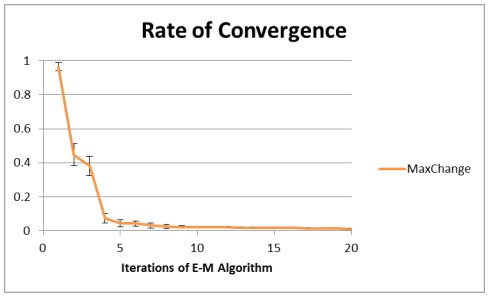


Fig. 4: The convergence rate of our algorithm.

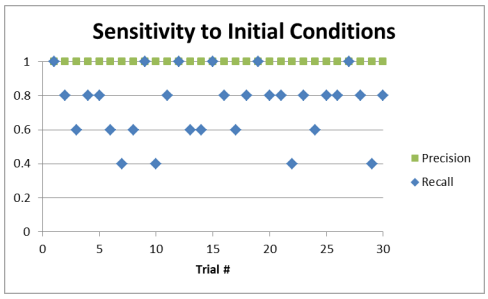


Fig. 5: Results from running 30 independent trials on the same dataset with random initial parameter settings.

consider using an ensemble of randomly-initialized instances of EM, predicting that a link exists if it is determined to be present by any one of the instances in the ensemble.[1] In Figure 6, we see that an ensemble of size ten improved recall from 82% to 90% while maintaining greater than 99% precision but that further increasing the size of the ensemble yielded diminishing marginal benefit.

D. Robustness of Model Assumptions

Our implementation of our algorithm fits the distribution of inter-event times for spontaneously generated events from each entity and the distribution of lag times between causal events to exponential distribution models. In Table I we evaluate the effectiveness of our algorithm when the processes driving network activity deviate significantly from that model. In particular, we run experiments using synthetic network data generated according to Exponential, Pareto (power-law),

[1]We chose this voting strategy because of the low false positive rate observed in Section III-B. Different strategies may be appropriate in other contexts.

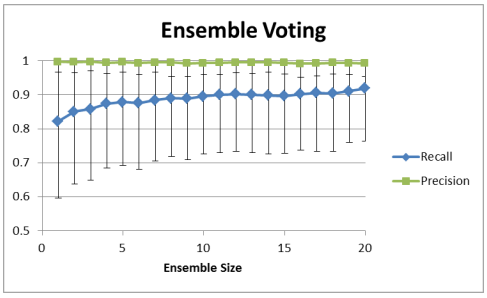


Fig. 6: Recall and precision of our algorithm using an ensemble of EM instances.

True Distribution	Recall	Precision
Exponential	0.812	0.998
Pareto	0.782	0.923
Log-Normal	0.694	0.892

TABLE I: Recall and precision of our algorithm when our model assumptions do not match the data.

and Log-Normal distribution models. We find that although performance degrades when our modeling assumptions do not hold, our algorithm is still able to achieve 78% recall and 92% precision for Pareto-generated data and 69% recall and 89% precision for Log-Normal-generated data, both models which differ significantly from the Exponential distribution.

E. Runtime Analysis

Using the heuristic described in Section II-E, computing the probability distribution over candidate triggers for an event requires a constant amount of computation for the most recent preceding event from each other entity, as well as for itself. Summing over all events by entity u yields a total of $O(|\mathcal{U}| \cdot |\mathcal{E}_u|)$, where $\mathcal{E}_u$ is the set of events from u. Computing the updated parameter estimates for g_u, l_u, and the ρ values require the same. For data generated synthetically as described in Section III-A with parameters n, k, l, g, and d, the expected number of events for entity u is $(k+1) \cdot d/g$, yielding an expected runtime of $O((n+1) \cdot (k+1) \cdot d/g)$. We validate these results with experiments on synthetic data. The empirical results shown in Figure 7 indicate that the runtime is linear in n, k, d, and $1/g$ and constant with respect to l, which agrees with our theoretical analysis.

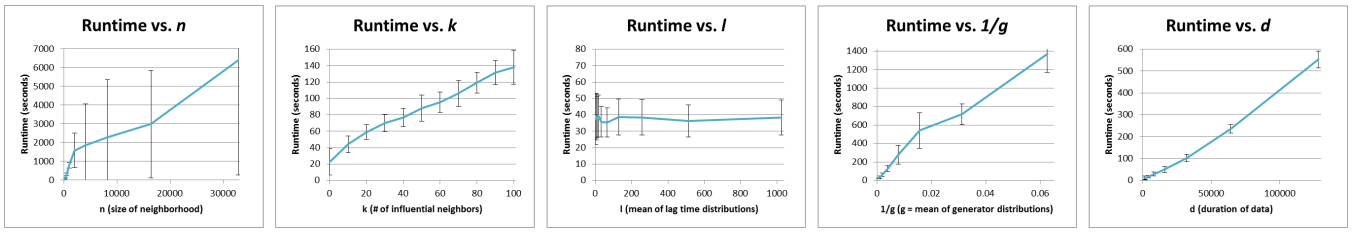


Fig. 7: Dependence of runtime performance on various network parameters.

IV. Future Work

There are several promising directions for future work. Fox et al. [2] demonstrate that incorporating diurnal and weekly patterns into their model can improve results; it would be interesting to see whether the same is true for our work. Our model could be expanded to allow different types of events, for example communication events versus attack events, or to include a spatial element. Applying our algorithm to data from the same network over different time intervals could help study how influences change over time. We also plan to explore how the probabilistic trigger assignment output by our algorithm can be utilized for further analysis, for example identifying common motifs in flow paths. In contexts where additional content or meta-data is available, a hybrid approach combining our algorithm with existing methods may yield better results than either approach by itself. The minimal data requirements of our approach make it easy to adapt to new application domains as the need arises.

V. Acknowledgments

We would like to thank Cris Moore, Aaron Clauset, Mason Porter, and Ananthram Swami for fruitful discussions.

References

[1] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[2] E. W. Fox, M. B. Short, F. P. Schoenberg, K. D. Coronges, and A. L. Bertozzi. Modeling email networks and inferring leadership using self-exciting point processes. Submitted, 2014.

[3] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[4] P. F. Halpin and P. De Boeck. Modelling dyadic interaction with Hawkes processes. *Psychometrika*, 78(4):793–814, 2013.

[5] A. G. Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90, 1971.

[6] R. Karp. Reducibility among combinatorial problems. In R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer US, 1972.

[7] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 497–506, New York, NY, USA, 2009. ACM.

[8] A. Veen and F. P. Schoenberg. Estimation of space-time branching process models in seismology using an EM-type algorithm. *Journal of the American Statistical Association*, 103(482):614–624, 2008.

[9] G. Ver Steeg and A. Galstyan. Information-theoretic measures of influence based on content dynamics. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 3–12, New York, NY, USA, 2013. ACM.