Bayesian Inference of a Social Graph with Trace Feasibility Guarantees

Effrosyni Papanastasiou, Anastasios Giovanidis

Bayesian Inference of a Social Graph with Trace Feasibility Guarantees

Effrosyni Papanastasiou
Sorbonne University, CNRS, LIP6
F-75005, Paris, France
effrosyni.papanastasiou@lip6.fr

Anastasios Giovanidis
Sorbonne University, CNRS, LIP6
F-75005, Paris, France
anastasios.giovanidis@lip6.fr

***Abstract*—Network inference is the process of deciding what is the true unknown graph underlying a set of interactions between nodes. There is a vast literature on the subject, but most known methods have an important drawback: the inferred graph is not guaranteed to explain every interaction from the input trace. We consider this an important issue since such inferred graph cannot be used as input for applications that require a reliable estimate of the true graph. On the other hand, a graph having trace feasibility guarantees can help us better understand the true (hidden) interactions that may have taken place between nodes of interest. The inference of such graph is the goal of this paper. Firstly, given an activity log from a social network, we introduce a set of constraints that take into consideration all the hidden paths that are possible between the nodes of the trace, given their timestamps of interaction. Then, we develop a non-trivial modification of the Expectation-Maximization algorithm by Newman [1], that we call `Constrained-EM`, which incorporates the constraints and a set of auxiliary variables into the inference process to guide it towards the feasibility of the trace. Experimental results on real-world data from Twitter confirm that `Constrained-EM` generates a posterior distribution of graphs that explains all the events observed in the trace while presenting the desired properties of a scale-free, small-world graph. Our method also outperforms established methods in terms of feasibility and quality of the inferred graph.**

***Keywords*—social graph, network inference, network reconstruction, expectation maximization**

I. Introduction

Network inference, or *reconstruction* is the problem of predicting the presence or absence of edges between a set of nodes that form the vertices of a graph, given an observed set of data, i.e., the *trace* [2]. Network inference has long been considered an important task; initially, it attracted a lot of attention in computational biology with various works reconstructing biological networks using representation learning, Bayesian networks, etc [3]. At the same time, studies applying network inference on social network data began emerging in literature and continue until today, thanks to the rapid growth of Online Social Networks (OSNs) [4]–[6]. The main goal of such works is to infer the *influence* between users - an important property of a social network [7] - and usually rely on diffusion models that capture the way information is diffused through the network. Such models include the *Independent Cascade (IC)* model, also called *Susceptible-Infected-Recovered (SIR)* in epidemiology [8], [9] and the *Susceptible-Infected (SI)* model [10]. The *Linear* or *General Threshold* model has also been extensively used for network inference [7].

After the choice of a diffusion model, the corresponding model parameters can be learned with several approaches, such as maximum likelihood [7], [11], Expectation-Maximization (EM) [1], [8], [9] and other static or continuous-time models [7]. For example, Saito et al. [8], used the IC model and applied the EM algorithm to learn the pairwise transmission probabilities of influence between users. More recently, Bourigault et al. [9] presented an embedded version of the IC model on OSNs that learns information diffusion probabilities along with the representation of users in the latent space. More recently, Newman [1] proposed an EM algorithm that is designed for network inference given unreliable data with the help of a set of parameters that estimate the size of the errors. Peixoto [12] approached network reconstruction similarly to Newman and combined it with community detection, proving that the task of inferring edges can improve the accuracy of community detection and vice-versa.

In the aforementioned inference procedures, we observed that the *feasibility* of the trace in relation to the inferred network is not always guaranteed. This means that the inferred network may not accurately and completely *explain* the input trace. For example, the application of the algorithm proposed by Saito et al. [8] applied on a Twitter cascade of user tweets and retweets, showed that, while it can predict high diffusion probabilities for some pairs of users and, thus, explain why we observe some retweets in the trace, it cannot do so for every original tweet. As a result, for a considerable number of retweets, we do not know the source of their influence and we, therefore, cannot explain their existence in the trace. The same is true for Newman's network reconstruction algorithm [1].

In cases like the above, we say that the trace is not *feasible* with respect to the inferred network. We argue that the concept of trace feasibility is an important condition that an inference framework must meet if we wish to understand the produced networks with relation to the trace and get them to explain every instance of our data. Therefore, we approach the problem of network inference in a novel way by developing a method that infers a posterior distribution of feasible networks that

This work is funded by the ANR (French National Agency of Research) by the "FairEngine" project under grant ANR-19-CE25-0011.

can accurately explain the given trace while respecting the *temporal order* of the observed events (e.g., posts). Towards this goal, we propose a non-trivial modification of the EM inference procedure developed by Newman [1] by introducing a set of constraints that take into consideration all the (yet unknown) paths that are possible according to the timestamps of interaction between the nodes and therefore derive feasible graphs.

Our algorithm works for OSN traces that include hundreds of thousands to millions of nodes. We focus the analysis on data from Twitter but we could apply the constraint set on other domains as well. However, to do so, the feasibility constraints should be adjusted each time to the type of data that we work with.

II. Environment

A. Assumptions on the environment

In the present paper, we will work with traces available from OSNs like Twitter or Weibo. On these platforms, a set of users can generate content that we call *posts*. For example, in the case of Twitter, users can post either *original posts* (i.e., tweets) or *reposts* (i.e., retweets) of original posts from other users. For this paper, we make three crucial assumptions:

1) Users repost from users they follow, i.e., their *followees*.
2) The followee from whom each user reposts is included inside the available trace.
3) Each user can repost the same post only once.

Of course, the above do not always hold in reality, but they serve as a simplification for our work. As an extension, our method could be modified to include cases where users repost outside the available trace or users, or from users who are not even their followees (e.g., if they traced a tweet from trending topics, hashtags, search, etc).

For the diffusion of posts, we choose the SI diffusion model from epidemiology [10]: with respect to each post, a user can transition from the susceptible state (i.e., when one of their followees posts or reposts something) to the infected state (i.e., when the user reposts it) only once and cannot transition back. In addition, we consider that an infected user can influence their yet uninfected followers during all the consecutive timestamps.

B. Trace description

Throughout this paper, we use $\mathcal{P}$ to denote our *trace* that is a log of T *posts* and *reposts*, generated by a set $\mathcal{U}$ of $|\mathcal{U}|$ =N users on an OSN (N $\leq$ T). Each line in $\mathcal{P}$ is a quadruple (pid, t, uid, rid) that includes four types of information: i) the unique post id pid; ii) its timestamp t; iii) the $uid \in \mathcal{U}$ of the user who posted it; and iv) a repost id rid that is either equal to -1 if the post is an original post, or equal to the pid of the original post if it is a repost. All posts in $\mathcal{P}$ are ordered according to their timestamps. It is important to underline here that social media logs, like in Twitter do not provide the identity of the user from whom someone found and reposted a post; they only include the rid, from which we can then track the original author.

III. Problem formulation

A. Problem definition

Since posts propagate by reposting, we assume the existence of an underlying friendship network between the N different users in the trace that is unknown to us. This network is a directed *social graph* $G = (V, E)$ where the nodes are the users of the trace ($V = \mathcal{U}$) and the edges include the friendships between the users. We represent G with an adjacency matrix N×N, denoted by $\mathbf{A}$, where each element A_{ij} is equal to 1 if user j follows user i and 0 otherwise. Our intuition is that if a user j shares content frequently from user i, it is more probable that j follows i. The goal of this work is to infer the unknown friendship network G, with trace feasibility guarantees, by inferring the hidden path each original post takes from user to user in the trace. It is important to note here that we can only retrieve friendships between users that have had at least one interaction with each other in the given trace. The richer the trace, the more complete our network inference will be.

B. Preliminaries

Our method relies on rich information extracted from a social media trace during the pre-processing phase. We begin by extracting from $\mathcal{P}$ the set of original posts denoted by $\mathcal{S}$ with cardinality $|\mathcal{S}|$ = S. We denote by r_s the uid of the user who originally posted each post s, i.e., $(s, t, r_s, -1) \in \mathcal{P}$ for some t.

Definition 1 (Episode): For each original post $s \in \mathcal{S}$ we define an *episode* as a set of users $\mathcal{E}_s = r_s \cup \{u \in \mathcal{U} | \exists (pid, t) : (pid, t, u, s) \in \mathcal{P}\}$. The whole set of episodes is denoted by $\mathcal{E}$ and includes S episodes in total.

Each episode $\mathcal{E}_s$ includes the user who originally posted s, denoted by r_s, followed by the users who reposted it, in chronological order. We use $i \overset{s}{\prec} j$ to say that user i appears in $\mathcal{E}_s$ before j, and we call this pair a temporally *ordered pair* $(i, j)_s$. We count M_{ij} out of the S total episodes where $i \overset{s}{\prec} j$. If $M_{ij} > 0$, then it is possible that j has reposted from i an original post or repost and we call this pair an *active pair*. Hence, it is a very important quantity for the inference of the hidden post propagation paths and we will make use of it in the next sections. The total number of active pairs is denoted by L and is equal to $\sum_{i \neq j} \mathbf{1}(M_{ij} > 0)$, where $\mathbf{1}(z) = 1$ when z is true. All information extracted from the trace is summarized in Table I.

C. Diffusion model per episode

Given an episode $\mathcal{E}_s$ and an ordered pair $(i, j)_s$, we first define the value $X_{ij}(s) \in \{0, 1\}$ that is equal to 1 if user j reposted s directly from i (i.e., post s propagated from i to j) and equal to 0 otherwise. However, the real value of $X_{ij}(s)$ is still unknown to us. Therefore, for the given post s, if we look into the temporal order of its reposts, we could think of several feasible ways or *paths* through which the post could have spread to reach the users that reposted it. These paths form a *propagation graph* $G_s = \{V_s, E_s\}$ per episode, with

TABLE I
TRACE INFORMATION

Symbol	Definition
$\mathcal{P}$	Set of posts and reposts in the trace, $\|\mathcal{P}\| = \mathrm{T}$
$\mathcal{S}$	Set of original posts in the trace, $\|\mathcal{S}\| = \mathrm{S}$
$\mathcal{U}$	Set of users that posted or reposted a post
N	Number of different users appearing in the trace
$\mathcal{E}$	Set of episodes, $\|\mathcal{E}\| = \mathrm{S}$
$\mathcal{E}_s \in \mathcal{E}$	Episode of post s, $1 \leq s \leq \mathrm{S}$
r_s	The id of the user that originally posted s
$i \overset{s}{\prec} j$	User i reposted or posted s before j
M_{ij}	Number of episodes where $i \overset{s}{\prec} j$
L	Number of active pairs with $\mathrm{M}_{ij} > 0$

the users in episode $\mathcal{E}_s$ as nodes ($V_s = \mathcal{E}_s$), and the edges set E_s containing the edges that are activated for the given post. Each activated edge follows the direction of propagation, e.g., an edge (i, j) in G_s means that $X_{ij}(s) = 1$.

As mentioned before, it is crucial that the paths in G_s take into account the temporal order of the users' posts and reposts. For example, we can think of a hypothetical episode $\mathcal{E}_s$ of post s, depicted in Fig. 1. The user *Frosso (Fr)* was the first who posted s, followed by *Phoebe (Ph)* and then, *Anastasios (An)*. Therefore, the possible propagation graphs, based on the chronological ordering of each repost in the episode, are the following:

- Users *Phoebe* and *Anastasios* reposted it both directly from *Frosso*. This corresponds to a propagation graph G_s with two directed paths: i) from the node *Frosso* to the node *Phoebe*; and ii) from the node *Frosso* to the node *Anastasios* (case I in Fig. 1).
- *Phoebe* reposted it from *Frosso* and then *Anastasios* from *Phoebe*. This corresponds to a propagation graph G_s with one directed path: from *Frosso* to *Phoebe* and then to *Anastasios* (case II in Fig. 1).

Notice that each path in each possible tree follows the time-ordering of reposts. In addition, G_s is an *arborescence* with root r_s [13]: this means that (i) G_s is a DAG; and (ii) given the vertex r_s called the root and any other vertex u in G_s, there is exactly one directed path from r_s to u. Equivalently, G_s is a directed, rooted tree in which all edges point away from the root.

IV. FEASIBILITY AND CONSTRAINTS

A. Feasibility definition

Given our problem definition, for each original post in $\mathcal{S}$, we need to infer a propagation DAG that is *feasible* in relation to the trace.

Definition 2 (Feasible propagation DAG per episode): Given an episode $\mathcal{E}_s$ from the trace, we say that a propagation DAG $G_s = \{V_s, E_s\}$ is *feasible in relation to* $\mathcal{E}_s$, if $V_s = \mathcal{E}_s$ and there exists (at least) one directed path from the root user r_s to every other user $j \in V_s \backslash r_s$. For each edge (i, j) of the path it holds that $i \overset{s}{\prec} j$ in the trace.

Using a DAG for each episode, G_s with $s = 1, 2, .., \mathrm{S}$, we can construct the full adjacency matrix $\mathbf{A}$ of the final friendship graph G as follows: we set $A_{ij} = 1$ if there exists at least one propagation DAG G_s where the edge (i, j) exists, and 0 otherwise.

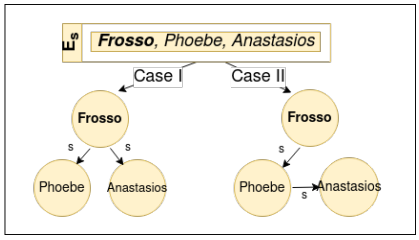


Fig. 1: Possible propagation graphs for episode $\mathcal{E}_s$.

Definition 3 (Feasible friendship graph): We define the adjacency matrix $\mathbf{A}$ of an inferred network G as *feasible* in relation to an OSN trace if, for every original post s there exists a subgraph in G, which is a feasible propagation graph G_s as defined in Def. 2.

For example, in Fig. 2, given a trace of three episodes $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3\}$ that involves five users *Frosso (Fr), Anastasios (An), Phoebe (Ph), Rachel (Ra), Joey (Jo)*, we can see the case of a non-feasible G and a feasible G'. G is non-feasible since there exists no feasible path from source node *Frosso* to *Anastasios* for the case of episode $\mathcal{E}_1$ and there exists no feasible path from source node *Anastasios* to *Frosso* for the case of episode $\mathcal{E}_3$. In contrast, G' is feasible since there exists a feasible propagation graph for each episode.

B. Feasibility constraints on retweets behavior

The real value of $X_{ij}(s)$ defined in Section III is not available, but we can limit the possible combinations by introducing a set of constraints on $X_{ij}(s)$, that ensure that a user $j \in \mathcal{E}_s$ has reposted s directly by at least one user $i \in \mathcal{E}_s$ who has reposted s earlier according to the trace. As a result, for each episode $\mathcal{E}_s \in \mathcal{E}$, and each user $j \in \mathcal{E}_s \backslash \{r_s\}$, the constraints take the following form:

$$\sum_{i \in \mathcal{E}_s \text{ s.t. } i \overset{s}{\prec} j} X_{ij}(s) \geq 1, \forall j \in \mathcal{E}_s \backslash \{r_s\}. \quad (1)$$

For all episodes we will get T-S constraints in total, i.e., as many constraints as the number of reposts with $\sum_{s=1}^{\mathrm{S}} (|\mathcal{E}_s| - 1) \cdot (|\mathcal{E}_s|)/2$ unknown variables in total. The way we formed the constraints, we allow the possibility that a user j has reposted the post s from more than one users instead of only one user for reasons that will become clear in the following section. For example, for episode $\mathcal{E}_s$ in Fig. 1 we will get constraints: $X_{FrossoPhoebe}(s) \geq 1$ and $X_{FrossoAnastasios}(s) + X_{PhoebeAnastasios}(s) \geq 1$. This means that the user $Phoebe$ has definitely reposted s from $Frosso$ and that $Anastasios$ has reposted it from $Frosso$, $Phoebe$, or both.

C. Diffusion probabilities

To be able to infer $X_{ij}(s)$ we make an important assumption: For every ordered pair $(i, j)_s$ in an episode $\mathcal{E}_s$, user j reposted post s from i independently of other episodes with unknown probability $\sigma_{ij} \in [0, 1]$ that is common for all episodes. In other words, $X_{ij}(s)$ is an independent Bernoulli random variable with mean σ_{ij} that is independent of s.

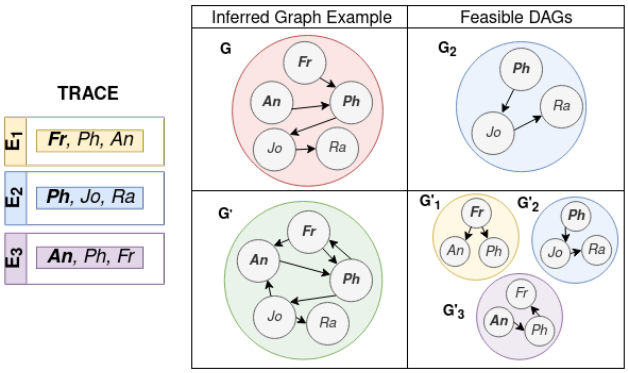


Fig. 2: Example of feasibility check given a trace.

Our choice to model the uncertainty about the path by using an independent Bernoulli random variable with a fixed mean σ_{ij}, assumes that user j does not have a contextual behavior that depends on the content of the episode, but rather behaves with randomness when choosing their sources of information. This serves as a simplification for the inference; as an extension, it could be further refined to include context-dependent mean values for different types of episodes. The value of σ_{ij} can be seen as the limiting frequency that user j reposts directly a post from i when the number of episodes goes to infinity. Given an ordered pair $(i,j)_s$, σ_{ij} is equal to:

$$\sigma_{ij} = \mathbb{E}\left[X_{ij}(s)\right]. \tag{2}$$

Given our intuition that the number of times a user j reposts a user i is indicative of their friendship we introduce to the problem the quantity Y_{ij}, as the (unknown) number of times that j reposted from i, out of the M_{ij} possible ones (the number of times they appear as an active pair). That is:

$$Y_{ij} = \sum_{m=1}^{M_{ij}} X_{ij}(s). \tag{3}$$

Since Y_{ij} is the sum of M_{ij} independent Bernoulli random variables with mean value σ_{ij}, Y_{ij} is an independent Binomial random variable with mean value $M_{ij}\sigma_{ij}$. That is:

$$\mathbb{E}[Y_{ij}] = \sum_{m=1}^{M_{ij}} \mathbb{E}[X_{ij}(s)] = \sum_{m=1}^{M_{ij}} \sigma_{ij} = M_{ij}\sigma_{ij}. \tag{4}$$

The value of σ_{ij} plays an important role in the inference of the relationship between i and j, as we will demonstrate in the following sections.

1) Constraints on diffusion probabilities σ_{ij}: Having transformed the problem from solving over $X_{ij}(s)$ to solving over σ_{ij}, we can constrain $\sigma_{ij} = \mathbb{E}\left[X_{ij}(s)\right]$ to be inside a specific set of values. If we take the expectation of the constraints in (1), for each episode $\mathcal{E}_s \in \mathcal{E}$, and each user $j \in \mathcal{E}_s \backslash \{r_s\}$, we end up with the following set of constraints on parameters σ_{ij}:

$$\sum_{i \in \mathcal{E}_s \text{ s.t. } i \overset{s}{\prec} j} \sigma_{ij} \geq 1, \forall j \in \mathcal{E}_s \backslash \{r_s\} \tag{5}$$

$$\sigma_{ij} \in [0,1], \ \forall (i,j) \in \mathcal{U}. \tag{6}$$

We define with $\mathrm{F}_{\boldsymbol{\sigma}}$ the feasibility space of the parameters vector $\boldsymbol{\sigma}$ that includes all σ_{ij} parameters, $(i,j) \in \mathcal{U}$, such that (5) and (6) hold. In this case, for episode $\mathcal{E}_s$ in Fig. 1 the constraints change to the following: $\sigma_{FrossoPhoebe} \geq 1$ and $\sigma_{FrossoAnastasios} + \sigma_{PhoebeAnastasios} \geq 1$. This means that user $Phoebe$ has reposted s from $Frosso$ with probability $\sigma_{FrossoPhoebe} = 1$ and that the probabilities of $Anastasios$ reposting it from $Frosso$ and $Phoebe$ must sum up to a value inside the interval $\in [1,2]$.

As a result, the parameters $\sigma_{ij} \in [0,1]$ are the problem unknowns that replace the $X_{ij}(s) \in \{0,1\}$ for all episodes where an action from user i precedes an action from j. For the whole trace $\mathcal{E}$, we will get a set of constraints $\mathcal{C} = \{c_1, c_2, ..., c_{(\text{T-S})}\}$, where, each element $c_k \in \mathcal{C}, 1 \leq k \leq$ (T-S), corresponds to the constraint of a (r_s, j) tuple, where $r_s \in \mathcal{E}_s$ and user $j \in \mathcal{E}_s \backslash \{r_s\}$, and is defined by (5)-(6). By imposing this set of constraints on the parameters σ_{ij}, we have drastically reduced the number of our problem's unknowns to the number of possible (i,j) pairs from the users set $\mathcal{U}$, i.e., we now have $\mathrm{N(N-1)}$ unknowns.

2) Removing redundant constraints: We notice that given a trace, some constraints become redundant and can be removed according to the following rules:

- If all parameters σ_{ij} that are included in a constraint $c_k \in \mathcal{C}$, are also included in a different constraint $c_w \in \mathcal{C}$, then c_w is removed from $\mathcal{C}$.
- In (5), we observe that the first constraint of each episode includes only one variable, which is the σ_{ij} between the first user $i = r_s$ and the second user j in the episode. Therefore, given also that $\sigma_{ij} \in [0,1]$, all parameters between the first and the second user of each episode become $\sigma_{ij} = 1$. As a result, the first constraint per episode is removed, since the solution for these parameters has already been found.

Note that for $M_{ij} = 0$, $\sigma_{ij} = 0$. Generally, the exact number of constraints by which our problem will be reduced depends on the characteristics of each trace.

V. Problem Modeling and Learning Method

As mentioned in the introduction, we develop a non-trivial modification of Newman's EM algorithm proposed in [1] that was designed for network inference given erroneous data. In our case, our data is not erroneous, but rather incomplete; however, we take advantage of Newman's probabilistic modeling and we adapt its parameters and the EM equations to our case. The code for our method `Constrained-EM` is publicly available and can be found at https://github.com/frossopap/constrained-em.

A. Parameters

Firstly, in a similar fashion to Newman [1], we assume that the relationship between the underlying network G and the trace can be expressed in the form of a probability function $P(\text{data}|\mathbf{A},\theta)$, which is the probability of generating the particular trace $\mathcal{P}$, given the adjacency matrix $\mathbf{A}$ and a set

of additional model parameters, denoted by θ. The parameters θ, added to cover a larger range of possibilities for the type of graph and the way the data is generated, are the following:

1) To model our uncertainty about the structure of the graph G, we assume a uniform prior probability ρ of the existence of an edge in any position between any pair of nodes, i.e. G has been drawn under the Erdős–Rényi model with parameter ρ.
2) The values σ_{ij}, which is the fixed probability that j shares content from i.
3) The *true-positive utilization rate* α: the probability of post propagation through existing edges of the underlying network G.
4) The *false-positive utilization rate* β: the probability of post propagation through non-existing edges of the underlying network G.

We see that α and β are global parameters, conditioned on the existence or not of an edge in the ground truth network G.

B. Learning Method

To find the most probable value of the parameters θ given the observed data and infer a graph with maximum likelihood, we will develop an application of Expectation-Maximization (EM): an iterative algorithm designed to find the maximum a posteriori (MAP) estimates of parameters in statistical models that depend on unobserved latent variables. Each EM iteration will alternate between two steps: i) an expectation (E) step, which creates the expectation of the log-likelihood using the current estimate for the parameters θ; and ii) a maximization (M) step, which finds the parameters that maximize the expected log-likelihood of the E-step. The estimated parameters are then used in the next E-step and so on until convergence is reached.

We begin in the same way as Newman [1] and we apply the Bayes' rule:

$$P(\mathbf{A},\theta|\text{data}) = \frac{P(\text{data}|\mathbf{A},\theta)P(\mathbf{A}|\theta)P(\theta)}{P(\text{data})}. \tag{7}$$

The probability that we get the specific set of reposts, given $\mathbf{A}$ and the parameters $\theta = \{\alpha, \beta, \rho, \boldsymbol{\sigma}\}$, differs here from Newman since we have introduced the hidden number of interactions between users, Y_{ij}. Given the ordered nodes of an episode, each repost path is chosen independently per episode. In addition, we assumed as prior knowledge that between any two nodes in $\mathbf{A}$ an edge has been drawn with probability ρ. Therefore we get:

$$P(\text{data}|\mathbf{A},\theta)P(\mathbf{A}|\theta) = \prod_{i\neq j}\Big[\alpha^{Y_{ij}}(1-\alpha)^{M_{ij}-Y_{ij}}\rho\Big]^{A_{ij}} \times \Big[\beta^{Y_{ij}}(1-\beta)^{M_{ij}-Y_{ij}}(1-\rho)\Big]^{1-A_{ij}}. \tag{8}$$

Given this type of modeling, when $A_{ij} = 1$, the Y_{ij} out of the M_{ij} experiments are successful, each with probability α. When $A_{ij} = 0$, the Y_{ij} out of M_{ij} experiments are successful, each with probability β. For the whole set of parameters θ, we assume a uniform prior probability $P(\theta)$. If we sum (7) over all possible networks $\mathbf{A}$, we find that $P(\theta|\text{data}) = \sum_{\mathbf{A}} P(\mathbf{A},\theta|\text{data})$. Then, as suggested by Newman [1], we can apply the well-known Jensen's inequality on the log of $P(\theta|\text{data})$:

$$\log P(\theta|\text{data}) = \log\sum_{\mathbf{A}} P(\mathbf{A},\theta|\text{data}) \geq \sum_{\mathbf{A}} q(\mathbf{A}) \log\frac{P(\mathbf{A},\theta|\text{data})}{q(\mathbf{A})} \tag{9}$$

where $q(\mathbf{A})$ is any probability distribution over networks $\mathbf{A}$ satisfying $\sum_{\mathbf{A}} q(\mathbf{A}) = 1$. We also define the posterior probability of an edge existing between i and j by $Q_{ij} = P(A_{ij} = 1|\text{data},\theta) = \sum_{\mathbf{A}} q(\mathbf{A})A_{ij}$.

For the E-step, we modify the Newman algorithm by taking the expectation over the set of random variables Y_{ij} at both sides of (9):

$$\begin{aligned}\mathbb{E}[\log P(\theta|\text{data})] &\geq \mathbb{E}[\sum_{\mathbf{A}} q(\mathbf{A}) \log\frac{P(\mathbf{A},\theta|\text{data})}{q(\mathbf{A})}] \\ &= \sum_{\mathbf{A}} q(\mathbf{A})\big(\mathbb{E}[\log P(\mathbf{A},\theta|\text{data})] - \log q(\mathbf{A})\big).\end{aligned} \tag{10}$$

To find $\mathbb{E}[\log P(\mathbf{A},\theta|\text{data})]$, we replace (8) into (7). Setting $\Gamma = P(\theta)/P(\text{data})$, the expectation of the log of (7) becomes:

$$\begin{aligned}\mathbb{E}[\log P(\mathbf{A},\theta|\text{data})] = log\Gamma + \sum_{i\neq j}\Big[A_{ij}\Big(\log\rho + \mathbb{E}[Y_{ij}]\log\alpha + \\ + (M_{ij} - \mathbb{E}[Y_{ij}])\log(1-\alpha)\Big) + (1-A_{ij})\Big(\log(1-\rho) + \\ + \mathbb{E}[Y_{ij}]\log\beta + (M_{ij} - \mathbb{E}[Y_{ij}])\log(1-\beta)\Big)\Big].\end{aligned} \tag{11}$$

Then, by replacing (4) into (11), and then (11) into (10), we get:

$$\mathbb{E}[\log P(\theta|\text{data})] \geq \sum_{\mathbf{A}} q(\mathbf{A}) \log\frac{D_{ij}}{q(\mathbf{A})} \tag{12}$$

$$\begin{aligned}\text{where, } D_{ij} = \Gamma\prod_{i\neq j}\Big[\rho\alpha^{M_{ij}\sigma_{ij}}(1-\alpha)^{M_{ij}(1-\sigma_{ij})}\Big]^{A_{ij}} \\ \times\Big[(1-\rho)\beta^{M_{ij}\sigma_{ij}}(1-\beta)^{M_{ij}(1-\sigma_{ij})}\Big]^{1-A_{ij}}.\end{aligned} \tag{13}$$

For the M-step of the EM algorithm, the function that we want to maximize is $\mathbb{E}[\log P(\theta|\text{data})]$. To do so, we need to find the unknown values, $q(\mathbf{A})$ and $\theta = \{\alpha, \beta, \rho, \boldsymbol{\sigma}\}$, that maximize the expectation on the left-hand side of (12), under the feasibility constraints on the parameters set θ. From these, only the σ_{ij} have an important constraint set, specified in (5) and (6).

C. Solution

1) With respect to $q(\mathbf{A})$: We notice that the choice of $q(\mathbf{A})$ that achieves equality (i.e. maximizes the right-hand side) in (12) is:

$$q(\mathbf{A}) = \frac{D_{ij}}{\sum_{\mathbf{A}} D_{ij}}. \tag{14}$$

From (14), in a similar fashion to Newman's method [Eq. (13), 20], and because Γ cancels out, we get:

$$q(\mathbf{A}) = \prod_{i\neq j} Q_{ij}^{A_{ij}}(1-Q_{ij})^{1-A_{ij}} \tag{15}$$

where Q_{ij} is the posterior probability that there exists an edge between i and j:

$$Q_{ij} = \frac{\rho\alpha^{M_{ij}\sigma_{ij}}(1-\alpha)^{M_{ij}(1-\sigma_{ij})}}{\rho\alpha^{M_{ij}\sigma_{ij}}(1-\alpha)^{M_{ij}(1-\sigma_{ij})} + (1-\rho)\beta^{M_{ij}\sigma_{ij}}(1-\beta)^{M_{ij}(1-\sigma_{ij})}}. \tag{16}$$

The expression here is also different from Newman, since in the exponents we get the expected number of events (using $M_{ij}\sigma_{ij}$) instead of the number of times j reposts from origin i directly (that is provided directly by the data). Notice also that for $M_{ij} = 0$, Q_{ij} becomes equal to the prior probability ρ. Moreover, from (14) we observe that $q(\mathbf{A})$ is the posterior probability distribution over all possible networks $\mathbf{A}$, $P(\mathbf{A}, \theta|\text{data})$ when Y_{ij} is replaced by its expected value $M_{ij}\sigma_{ij}$.

2) With respect to σ_{ij}: Our goal now is to find the parameters θ that maximize the right-hand size of (12), given the maximising distribution for $q(\mathbf{A})$ in (14), hence given the values of Q_{ij} in (15). If we take into account that $Q_{ij} = \sum_{\mathbf{A}} q(\mathbf{A})A_{ij}$ and also that $\sum_{\mathbf{A}} q(\mathbf{A}) = 1$, by rearranging the right-hand side of (12), the problem becomes equivalent to maximizing:

$$\sum_{\mathbf{A}} q(\mathbf{A}) \sum_{i\neq j} \sigma_{ij} M_{ij} \left(A_{ij} \log\frac{\alpha}{1-\alpha} + (1-A_{ij}) \log\frac{\beta}{1-\beta} \right)$$
$$= \sum_{i\neq j} \sigma_{ij} M_{ij} \left(Q_{ij} \log\frac{\alpha}{1-\alpha} + (1-Q_{ij}) \log\frac{\beta}{1-\beta} \right). \tag{17}$$

Finally, if $\boldsymbol{\sigma}$ is a vector of size L that includes all σ_{ij} instances, we end up with the following linear optimization problem:

$$\max_{\sigma} \sum_{i\neq j} \sigma_{ij}(W_{ij} - \lambda) \tag{18}$$

s.t. $\boldsymbol{\sigma} \in F_{\boldsymbol{\sigma}}$

where $W_{ij} = M_{ij} \left(Q_{ij} \log\frac{\alpha}{1-\alpha} + (1-Q_{ij}) \log\frac{\beta}{1-\beta} \right)$

and $\lambda > 0$ some given penalty for regularisation. (19)

Our goal is to infer a graph that is feasible and also has the minimum possible number of edges; this is why we added the value λ as a penalty into the maximization goal per each iteration. Without it, all (i, j) pairs with $W_{ij} > 0$ would immediately get their $\sigma_{ij} = 1$, leading to the inference of more edges than necessary. Therefore, we choose to set λ equal to the largest W_{ij} value, i.e. $\lambda = \max_{(i,j)\in W} W_{ij}$. This choice of λ forces the optimization goal to be negative and thus, to be guided only by the provided constraints. It is equivalent to penalizing the total expected number of inferred edges.

3) With respect to α, β, ρ: Next, we maximize the right-hand side of (12) in terms of parameter α by differentiating it with respect to α and then setting it equal to zero (while holding σ_{ij}, q constant):

$$\sum_{i\neq j} Q_{ij} M_{ij} \left(\frac{\sigma_{ij}}{\alpha} - \frac{1-\sigma_{ij}}{1-\alpha} \right) = 0. \tag{20}$$

After rearranging, we get:

$$\alpha = \frac{\sum_{i\neq j} M_{ij}\sigma_{ij}Q_{ij}}{\sum_{i\neq j} M_{ij}Q_{ij}}. \tag{21}$$

Similarly for β and ρ, we get:

$$\beta = \frac{\sum_{i\neq j} M_{ij}\sigma_{ij}(1-Q_{ij})}{\sum_{i\neq j} M_{ij}(1-Q_{ij})}, \tag{22}$$

$$\rho = \frac{1}{\mathrm{N}(\mathrm{N}-1)} \sum_{i\neq j} Q_{ij} \tag{23}$$

where N is the number of total different users in the trace.

Finally, we end up with an iterative EM algorithm that iterates between finding an optimal value for q, i.e. a value that allows for (12) to hold with equality (E-step), and then holding it constant to maximize the right-hand side of (12) (and therefore also the expectation in the left-hand side of (12)) with respect to θ (M-step), through the updates in (18), (19), (21), (22), (23). Our algorithm converges when the L2 norm of improvement $||\mathbf{Q}_{new} - \mathbf{Q}_{old}|| < \epsilon$ falls under some threshold ϵ that we choose in advance, where $\mathbf{Q}$ is the matrix containing all the Q_{ij} values.

VI. Experimental Evaluation

A. Dataset

To evaluate our approach we use a real-world Twitter dataset coming from Kaggle, referred to as *Russian* [1]. It contains almost 2 million tweets and retweets emitted from 181,621 users during the Russian presidential elections of 2018. Users are anonymous and tweets are ordered in time. We choose the first 500,000 lines of the trace. Each line is a quadruple *[PostID, TimeStamp, UserID, RePostID]*. We remove all the tweets that have not been retweeted by any users and all the retweets for which we do not know the user who originally posted it. In addition, we delete retweets that appear more than once for the same user and tweet. The final statistics are summarised in Table II.

B. Experimental Settings

1) Environment: We run the experiments on a Google Cloud virtual instance with 16 vCPUs and 128 GB RAM. For the solution of the optimization problem, we use PuLP [2], an open-source linear programming library for Python.

2) Number of constraints: For a trace set of size $|\mathcal{P}| = 216,989$ the number of constraints is $198,543$. After removing the redundant ones according to Section IV-C2, we observe an approximately 10% percent decrease in the number of constraints ($= 170,209$). However, the level of decrease generally depends on the sparsity of the trace network itself. The more connected the initial network, the higher the decrease we observe.

3) Initialization and convergence rule: Parameters α, β and r are initialized randomly. The threshold ϵ of our algorithm's convergence criterion on the L2 norm $||\mathbf{Q}_{new} - \mathbf{Q}_{old}|| < \epsilon$ is set equal to $\epsilon = 0.001$.

[1] https://www.kaggle.com/borisch/russian-election-2018-twitter
[2] https://pypi.org/project/PuLP/

TABLE II
BASIC STATISTICS ON RUSSIAN AFTER PRE-PROCESSING

	Russian
Time window	20 days
Trace size $\lvert\mathcal{P}\rvert$	216,989
#original tweets	14,781
#retweets	202,208
#users	42,011
% users with #tweets >0	13.00
% users with #retweets >0	94.30

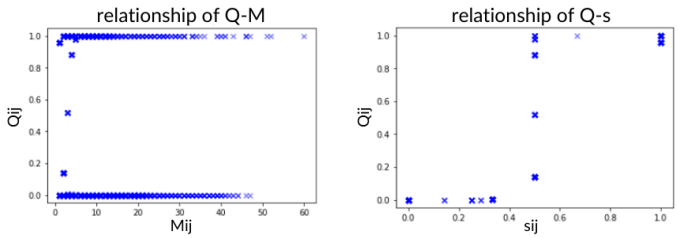


Fig. 3: `Constrained-EM` results.

VII. RESULTS AND COMPARISON

Our algorithm `Constrained-EM` takes 109 iterations and approximately 60 hours to converge. The converged parameters of our method, are $\alpha^* = 0.9932, \beta^* = 0.0001, r^* = 0.0034$. This means that there is an approximately 99% probability that a post propagated through an edge present in the inferred network G. The small value of β suggests that there are very few false-positive utilized edges: a post from the trace propagates through an edge where none exists around 0.001% of the time.

In Fig. 3 (left), we show the final values of Q_{ij} in relation to the number of times M_{ij} that each (i, j) edge is observed in the trace. As we can see, when M_{ij} is relatively small, Q_{ij} alternates between the whole range of $[0, 1]$. As M_{ij} becomes larger, the Q_{ij} is either 0 or 1 and finally, for the larger values of M_{ij}, the value of Q_{ij} stabilizes to 1. This could be attributed to the fact that the more times an edge is observed in a trace, the more certain we become about the existence of the edge; equivalently the more times a user retweets after some other user, the more certain we become about the existence of a friendship between them. Moreover, the different Q_{ij} results for the same M_{ij} values depict the important role the constraints play in the inference process. Regarding the σ_{ij} values, we observe in Fig. 3 (right) that for $\sigma_{ij} > 0.5$, Q_{ij} is almost 1 and for $\sigma_{ij} < 0.5$, Q_{ij} becomes 0. For $\sigma_{ij} = 0.5$, Q_{ij} alternates between values in $[0.1, 1]$. Hence, we confirm our intuition that the probability that user j follows i depends on the probability σ_{ij} that j reposts i.

A. Comparison

To generate the hidden friendship network G inferred by `Constrained-EM`, we round up the edges (i, j) with $Q_{ij} > 0.5$ to 1, and the edges with $Q_{ij} <= 0.5$ to 0. Each existing (i, j) edge suggests that user j follows user i. Then, we compare `Constrained-EM` with the following inference methods:

- **`Star`**: a baseline inference method that creates an edge from the user who originally posted each tweet s in the trace, to every other user who retweeted it.
- **`Chain`**: a baseline inference method which, for each episode $\mathcal{E}_s$ in the trace, creates a single long path between the user nodes in $\mathcal{E}_s$ according to the timestamps of their actions: from the user who originally posted s, to the user who retweeted s first, to the next user who retweeted s, and so on.
- **`Saito et al.` [8]**: an EM-based algorithm that considers the friendship graph as a pre-given and infers the influence probabilities k_{ij}. For evaluation, we create a graph by drawing an edge (i, j) whenever $k_{ij} > 0.5$.
- **`Newman` [1]**: the EM-based algorithm by Newman, presented in the introduction. It is not designed to consider hidden paths between user tweets and retweets and therefore infers networks that are not feasible. However, it would be useful to observe the differences with our method. For evaluation, we create a graph by drawing an edge (i, j) whenever the friendship probability Q_{ij} for a user pair (i, j) is greater than 0.5.

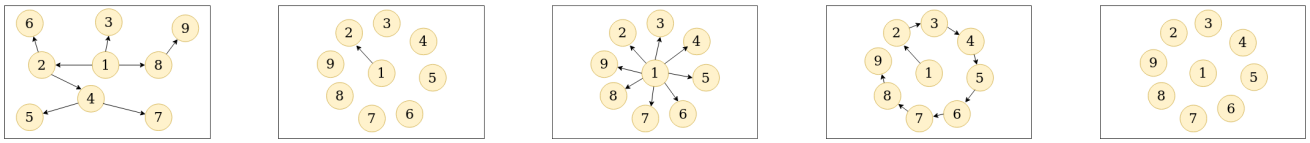
To evaluate each method, since the ground truth is not available, we count how many episodes in the trace are feasible, given the graph inferred by each method. Moreover, we examine to what extent the properties of each graph resemble those of a real network. Therefore, we first look into the propagation graph inferred by each method, given a random episode from the trace $\mathcal{E}_s = \{1, 2, ..., 9\}$ (users are anonymized by integers) and demonstrate it in Fig. 4. Then, we compare each method on different graph statistics (Table III). On top of that, we compare the in and out-degree complementary cumulative distribution functions (CCDFs) of each graph (Fig. 5). From these, we can make the following observations for each method:

1) `Star`: Fig. 4 shows that the graph inferred by `Star` explains the whole episode $\mathcal{E}_s$ by connecting the author directly to each user that retweeted it. This is repeated for all episodes in the trace, achieving 100% feasibility, as shown in Table III. However, the way nodes are connected is heuristic and untrustworthy. This is also reflected in the unrealistically high maximum out-degree of its graph ($= 4,524$), compared to the other methods.

2) `Chain`: Fig. 4 shows that the graph inferred by `Chain` explains the whole episode $\mathcal{E}_s$ with a 100% feasibility rate for all episodes. Again, `Chain` may return a feasible graph, but its high diameter ($= 183$) prevents us from choosing it as a real-world scenario.

3) `Saito et al.` [8]: In Fig. 4 we observe that for episode $\mathcal{E}_s$, the inferred graph explains only the retweet by user 2 (through the $(1, 2)$ edge). For the whole trace, it explains only 10% of it (Table III). Moreover, we observe that the graph has no strongly connected components (#Scc in Table III) which could also explain the small number of feasible episodes.

4) `Newman` [1]: The vanilla method by Newman that does not use any constraints, cannot explain any interaction in



(a) `Constrained-EM` (b) `Saito et al.` [8] (c) `Star` (d) `Chain` (e) `Newman` [1]

Fig. 4: Propagation graph inference for an episode $\mathcal{E}_s = \{1, 2, ..., 9\}$ from the trace.

TABLE III
INFERRED GRAPH METRICS FOR EACH METHOD

Graph Type	% Feasible Episodes	#Edges	Avg out-deg.	Max out-deg.	Max in-deg.	Diameter	Avg shortest path	#Scc
`Constrained-EM`	98.90	100,073	2.38	448	133	28	5.95	17
`Saito et al.` [8]	10.06	20,542	0.49	21	18	88	8.78	0
`Star`	100.00	162,570	3.87	4,524	173	18	6.03	23
`Chain`	100.00	194,964	4.64	224	252	183	6.66	12
`Newman` [1]	0.48	21,431	0.51	59	1,481	12	4.55	8

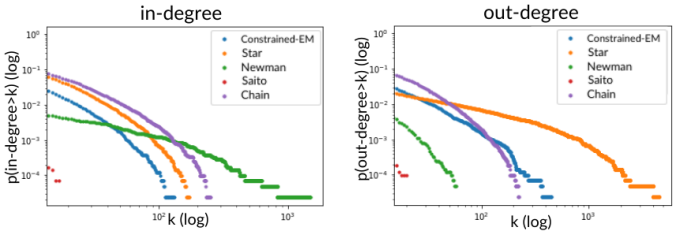


Fig. 5: CCDF for all methods (on a log-log scale).

episode $\mathcal{E}_s$, as expected. For the whole trace, it explains less than 0.50% of the episodes.

5) `Constrained-EM`: Fig. 4 shows that our method can explain the whole $\mathcal{E}_s$, while for all episodes it achieves an almost 99% feasibility for the given convergence rule (Table III). The remaining 1% that is left unexplained is attributed to the (rare) case when in an $\mathcal{E}_s$ there exist more than two users that have the same M_{ij} values and thus, are not distinguished with sufficient certainty ($\sigma_{ij} = 0.5$). We underline that our method is feasible with the least number of edges ($= 100,073$) compared to `Star` and `Chain`. Moreover, in Fig. 5, especially in out-degree CCDF, `Constrained-EM` presents a close to *scale-free* behavior, with both a heavy tail and an almost linear distribution line. On top of that, the average shortest path of our graph is close to 6. Given the well-known notion of *six degrees of separation*, or equivalently, the idea that in a *small-world* graph, any two pairs of nodes are separated by less than six nodes [14], we conclude that our graph has properties close to these of a scale-free, small-world network. Hence, we consider it to be a trustworthy framework for feasible network inference.

VIII. Conclusion and Future Work

As demonstrated above, given a log of tweets and retweets, our method `Constrained-EM` successfully infers a feasible friendship graph that explains each tweet's propagation from user to user, while being economical in the number of drawn edges. On top of that, we showed that our graph has properties that are close to these of a scale-free, small-world network. Therefore, `Constrained-EM` generates feasible graphs that are more reasonable than simple heuristics like `Star` and `Chain`. It is worth noting that our method could be applied on other domains where feasibility constraints can be imposed, such as epidemics, biology, etc. As future work, we plan to investigate ways in which `Constrained-EM` can be improved in terms of convergence speed.

References

[1] M. E. J. Newman, "Network structure from rich but noisy data", *Nature Physics*, vol. 14, 2018, pp. 67-75.

[2] J-P. Vert and Y. Yamanishi, "Supervised graph inference", in *Proceedings of the 17th International Conference on Neural Information Processing Systems*, 2004, pp. 1433–1440.

[3] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using bayesian networks to analyze expression data", in *Journal of Computational Biology*, vol. 7, pp. 601–620, 2000.

[4] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks", *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[5] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network", in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 137–146.

[6] M. E. J. Newman, "Clustering and preferential attachment in growing networks", *Physical Review Letters E*, vol. 64, no. 2, 2001.

[7] A. Goyal, F. Bonchi, and L. Lakshmanan, "Learning influence probabilities in social networks", in *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, 2010, pp. 241-250.

[8] K. Saito, R. Nakano, and M. Kimura, "Prediction of Information Diffusion Probabilities for Independent Cascade Model", in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, vol. 5179, 2008, pp. 67-75.

[9] S. Bourigault, S. Lamprier, and P. Gallinari, "Representation Learning for Information Diffusion through Social Networks: an Embedded Cascade Model", in *International Conference on Web Search and Data Mining*, 2016, pp. 573-582.

[10] D. J. Daley and J. Gani, *Epidemic Modelling: An Introduction*. Cambridge University Press, 1999.

[11] C. Lagnier, L. Denoyer, E. Gaussier, and P. Gallinari, "Predicting Information Diffusion in Social Networks using Content and User's Profiles", in *35th European Conference on IR Research*, 2013, pp. 74-85.

[12] T. P. Peixoto, "Network reconstruction and community detection from dynamics", *Physical Review Letters*, vol. 123, no. 12, 2019.

[13] G. Gordon, "A greedoid polynomial which distinguishes rooted arborescences", in *Proceedings of the American Mathematical Society*, 1989.

[14] A-L. Barabási, "Network Science Book", *Center for Complex Network Research, Northeastern University*, 2014.