Constrained Expectation-Maximisation for inference of social graphs explaining online user-user interactions

Effrosyni Papanastasiou, Anastasios Giovanidis

Constrained Expectation-Maximisation for inference of social graphs explaining online user-user interactions

Effrosyni Papanastasiou and Anastasios Giovanidis

Sorbonne University, CNRS, LIP6, F-75005, Paris, France
effrosyni.papanastasiou@etu.sorbonne-universite.fr, anastasios.giovanidis@lip6.fr

November 28, 2022

ABSTRACT

Current network inference algorithms fail to generate graphs with edges that can explain whole sequences of node interactions in a given dataset or *trace*. To quantify how well an inferred graph can explain a trace, we introduce *feasibility*, a novel quality criterion, and suggest that it is linked to the result's accuracy. In addition, we propose CEM-*, a network inference method that guarantees 100% feasibility given online social media traces, which is a non-trivial extension of the Expectation-Maximization algorithm developed by Newman (2018). We propose a set of linear optimization updates that incorporate a set of auxiliary variables and a set of feasibility constraints; the latter takes into consideration all the hidden paths that are possible between users based on their timestamps of interaction and guide the inference toward feasibility. We provide two CEM-* variations, that assume either an Erdős–Rényi (ER) or a Stochastic Block Model (SBM) prior for the underlying graph's unknown distribution. Extensive experiments on one synthetic and one real-world Twitter dataset show that for both priors CEM-* can generate a posterior distribution of graphs that explains the whole trace while being closer to the ground truth. As an additional benefit, the use of the SBM prior infers and clusters users simultaneously during optimization. CEM-* outperforms baseline and state-of-the-art methods in terms of feasibility, run-time, and precision of the inferred graph and communities. Finally, we propose a heuristic to adapt the inference to lower feasibility requirements and show how it can affect the precision of the result.

Key words. online social networks – network inference – network reconstruction – stochastic block model– expectation maximization

1. Introduction

Given a set of observed data, *network inference*, or *reconstruction* is the task of determining whether an edge exists or not between any pair of nodes that have interacted at some point in time. Network inference was first used in computational biology, where it was invented as a tool to recreate and explain complex interactions between important nodes, such as proteins or genes (Friedman et al. 2000). Network inference methods have since been applied in a variety of fields besides biology. Examples include epidemiology (Zhang et al. 2021; Firestone et al. 2020), finance (Giesecket et al. 2020), and telecommunications (Wu et al. 2022). The main goal of this paper is network inference in the domain of Online Social Networks (OSNs). Their enormous growth in the last decade has resulted in huge amounts of information circulating online from user to user. As a result, research has turned to inference algorithms to derive diverse types of networks which can be useful in various fields such as marketing, advertising, and politics. In advertising, for example, inference algorithms have been employed to derive the probabilities of influence between users, or the way that a specific news piece has diffused on the platform (Gomez-Rodriguez et al. 2012).

The reason why non-trivial methods such as network inference are needed to infer these types of networks is lying in the structure of the online datasets themselves. Regarding the diffusion of information through an online platform, the data we can find is limited and does not directly depict how it propagates from user to user. On Twitter, for example, given a tweet by an author and the users that retweeted it, we can get information such as the timestamps of each retweet, but we cannot know where they really retweeted it from[1]. This suggests that inferring the true propagation of a tweet when the friendship graph is unknown is not trivial. Network inference algorithms are thus brought into play and make it possible to infer the real way that information propagates on OSNs by exploiting the available interactions between users (the *trace*). Regarding the learning method itself, different methods have been employed, including maximum likelihood (Harris et al. 1998), expectation-maximization (EM) (Dempster et al. 1977), and other models of influence computation, such as Discrete-Time and Continuous-Time Models (Goyal et al. 2010).

When looking at the result of an inference method, one can check whether the input trace is what we call, *feasible*, given the generated network. We can do this by verifying that the inferred graph of connections includes a path from the author of every original post (e.g., tweet) to all other users that shared the post (e.g., via retweets) in the trace. For feasibility, this path should respect the chronological order of the respective interactions in the trace. However, as we will show later experimentally, existing works have disregarded feasibility as a quality criterion that the inferred graph must meet. Therefore, in this paper, we propose trace feasibility as an imperative requirement that must be met by an inference framework applied to OSNs.

[1] According to the Twitter API documentation of a Tweet Object, the "retweets of retweets do not show representations of the intermediary retweet, but only the original Tweet." https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet

Our intuition behind this proposal is that a feasible graph that can explain all the interactions and their chronological order inside the trace is closer to the real one. This could become more obvious if we think of what non-feasibility entails: suppose that there is an interaction by a user in the trace (e.g., reshare of a post) that cannot be explained by the inferred graph. This means that there is no path (with one or more hops) in the inferred graph from the user author to the user who reshared the post, or that the path is temporally not feasible. Then, either the latter user found this post from some other source (e.g., platform recommendation), or there is an error in the inference because the two users appear disconnected or connected in the wrong (temporally non-feasible) direction. By enforcing feasibility during graph inference, we guarantee that the graph can reproduce and explain all events and interactions observed in the available trace. Of course, in reality, a percentage of the observed interactions can come from indirect diffusion (e.g., recommendations); as we will show later, it is possible to take this into account by assuming some fixed percentage of direct diffusion during the inference process.

Given the above motivation, we can examine whether current methods in the literature infer graphs that guarantee feasibility. By looking into the seminal work of Saito et al. (2008), we see that the results suffer from the fact that it is not possible to identify the source of influence for a large number of retweets, and therefore their existence in the trace cannot be explained. Therefore, trace feasibility given the inferred network of influence is not achieved. In another fundamental work, Gomez-Rodriguez et al. (2012) proposed the NetInf method to infer the optimal network that most accurately explains a sequencing of interactions. However, they only give approximate solutions that, when applied to real-world data, are neither feasible nor accurate. More recently, Newman (2018) introduced an EM algorithm that is designed for network inference using unreliable data. As the algorithm does not consider that there are hidden paths between the users, the feasibility of the trace given the inferred network is not guaranteed. Building on Newman's work, Peixoto (2019) was the first to propose a method that performs network reconstruction together with community detection. However, as we will validate experimentally, despite being more precise than the methods above, the results suffer from slow convergence times and again, do not always guarantee feasibility which has an impact on precision. Therefore, as we can see, the inference methods that are currently available in the literature suffer from the fact that they do not explicitly guarantee the feasibility of the results. This is extremely critical since the resulting graphs infer edges that cannot confirm the trace itself. Additionally, as we will show later, each method presents other smaller issues that could have been avoided by enforcing the feasibility guarantees that we propose.

As a solution to the above, we introduce a fresh approach to network inference, which we call CEM-* (Constrained Expectation Maximization). It infers a posterior distribution of feasible underlying graphs that explain the provided social trace while respecting the chronological order of the interactions observed. Since the structure of the underlying graph is not known, the definition of a prior that enforces a structure to the posterior inferred graph is necessary. In this work, we will introduce two special cases of CEM-*: (i) CEM-er, which uses an Erdős–Rényi (ER) prior, and (ii), CEM-sbm, which uses the Stochastic Block Model (SBM). Besides, CEM-* can be adjusted accordingly to include other priors as well. All in all, we enrich the literature with the following contributions:

- We define social trace feasibility, and discuss its importance for network inference in the domain of OSNs. To guarantee feasibility, we devise a set of inequalities (constraints) to account for all the possible hidden paths given the timestamps of interaction between the nodes in the social trace (users).

- We propose CEM-*, a non-trivial extension of the Expectation-Maximization algorithm originally proposed by Newman (2018) that further incorporates the above set of feasibility constraints. Its main advantage is that it formulates inference as a linear optimization problem, making the task easier to compute. For the graph's unknown distribution, we start with an ER prior, following Newman's (2018) formulation, and call the method CEM-er (see also our conference version (Papanastasiou & Giovanidis 2021)).

- We introduce CEM-sbm, a variation of CEM-er that uses an SBM instead of an ER prior that is more realistic to the underlying structure of social graphs. On top of graph inference, CEM-sbm allows us to infer and assign users in communities simultaneously during optimization. Its main benefit against Peixoto (2019), except for guaranteeing feasibility, is that it is more scalable and easier to compute.

- We apply CEM-* on a synthetic social trace and compare the inferred graph against the ground truth. We also apply it on a real-world Twitter trace with almost 300,000 tweets and more than 1,600,00 retweets and compare the result against the real friendship graph that we have available. Extensive numerical evaluations of CEM-* against other baseline and state-of-the-art inference methods demonstrate the algorithm's ability to run on large graphs and trace sizes, which is not always guaranteed by the alternatives.

- We show that real-world traces are not always 100% feasible given the real graph that underlies them and we propose a technique with which we can tune CEM-* to adapt to lower feasibility requirements. We evaluate to what extent tuning the inferred graph's feasibility can infer edges with better accuracy.

The rest of this paper is organized as follows: in Section 2 we present related literature. In Section 3 we introduce the formulation of the problem. Section 4 presents the modeling of the problem and the learning method that we follow. Section 5 describes the datasets that we use and the methodology of the experiments. Sections 6, 7 and 8 show the results of the experiments and the comparison with other methods for the synthetic and the real-world traces respectively. Section 9 presents conclusions and future work. The code for both CEM-er and CEM-sbm is publicly available on GitHub[2].

2. Related literature

Graph inference. Numerous studies have proposed graph inference methods by simultaneously recovering influence probabilities between users. This is usually possible by observing users' infection timestamps from the available cascades of interactions. For example, Goyal et al. (2010) compute probabilities from a real social graph and a log of actions on Flickr using Continuous and Discrete Time Models with incremental equations.

[2] https://github.com/effrosyni-papanastasiou/constrained-em

He and Liu (2017) presented an approach that recovers a graph from a small number of cascade samples by utilizing the similarities between strongly linked diffusion graphs. A different line of work focuses on learning embeddings to perform the same inference task: for instance, Wang et al. (2019) suggested predicting information diffusion by learning user embeddings that capture unique characteristics both of the diffusion and the network. Later, Bourigault et al. (2016) 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. (Zhang et al. 2018) proposed a probabilistic generative model to learn information cascade embeddings that predict the temporal dynamics of social influence.

Graph inference with incomplete data. Additionally, many works consider that the observed cascades are incomplete or partially observed, which is frequently the case in real-world settings. This is why a diffusion model must be chosen along with the learning method to represent how we believe that information has been passed through the cascades. Wu et al. (2013) for instance, created an EM method that can tolerate missing observations in a diffusion process that follows the continuous independent cascade (CIC) model. Daneshmand et al. (2014) proposed an L_1-regularized maximum likelihood inference method for a well-known Continuous-Time diffusion model. Lokhov (2016) introduced an approximate gradient descent approach that estimates the influence parameters using gradients of the likelihood calculated via mean-field approximation and dynamic message passing. Their formulation makes the computation tractable, but the complexity of the gradients causes slow convergence.

Selecting a prior when the ground truth is unknown. Several link prediction methods extract future or missing links in datasets in which the underlying graph connecting the users is known (Saito et al. 2008, Bourigault et al. 2016, Lagnier et al. 2013, Jin et al. 2020, Peel et al. 2022). However, our goal differs from these types of problems since we have to infer links in a setting where the neighborhoods of the nodes are unknown. We must therefore select a prior structure that is close to the underlying network. For example, Le et al. (2018) selected the SBM as the underlying network structure, because of its simplicity and its ability to approximate real networks. Similarly, Peixoto (2019), used the degree-corrected SBM as a prior, motivated by its ability to inform link prediction when dealing with incomplete or erroneous data. In another example, Newman (2018) experimented with different kinds of priors, such as the random graph, the Poisson edge model, and the SBM.

Neural networks. In a more recent line of work, recurrent neural networks have been used to predict edges given probability distributions conditioned on temporal sequences of past knowledge graphs (Jin et al. 2020). Neural networks usually require the graph of nodes as input. However, in most social media network settings the friendship graph of user nodes is either not known or has not been published by the creators of the datasets. This makes the use of neural networks for inferring hidden edges more challenging. We leave the use of such methods for network inference when the underlying graph is unknown or incomplete as a future interesting task.

3. Problem formulation

3.1. Input data trace

As mentioned above, to infer an unknown network we must provide as input a trace of interactions between the nodes of interest. In this paper, as we focus on traces from OSNs, our goal is to infer friendship graphs by looking into the online interactions between users, and more specifically into the *posts* and the *reposts* that they generate. On Twitter, for example, this corresponds to the tweets and retweets that the users exchange.

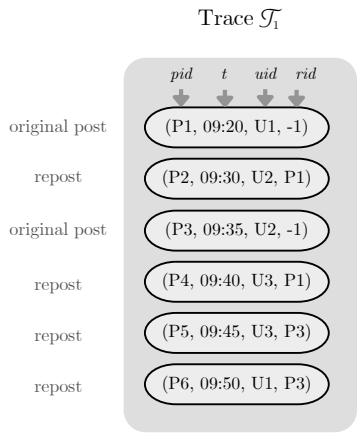


Fig. 1: Example of information available on an OSN trace.

Throughout the paper, we will use the following notation: the input interaction log with the posts and reposts is denoted by $\mathcal{T}$ and it includes T posts/reposts in total. For each instance in the trace, we keep only four types of information: its unique post id (*pid*), the time that the user posted it (t), the unique user id (*uid*), and the repost id (*rid*) that equals -1 if the post is original, or, if it is a repost, it is equal to some $pid \in \mathcal{T}$ which points to the original post instance in the trace. If a user is the author of a *pid* we mark them as $author_{pid}$. The set that includes all the users that participate in the trace is denoted by $\mathcal{U}$ and is of size $|\mathcal{U}| = N$. Figure 1 shows an example of a trace $\mathcal{T}_1$ like the one described above. It includes $T = 6$ posts/reposts instances and $N = 3$ users in total. The first instance in $\mathcal{T}_1$ is an original post with pid = P1, and is posted at t = 09:20 by author U1; the second instance with pid = P2, tells us that user U2 reposted at t = 09:30 the post with pid = P1 (mapped to the author U1), and so on.

3.2. Problem formulation

Given an available trace $\mathcal{T}$ of the activity of a set of users $\mathcal{U}$, we assume that there is an underlying friendship graph G connecting all users in $\mathcal{U}$ that is unknown and is what we are trying to infer. More formally, it is a directed friendship graph G where the nodes are the N users in $\mathcal{U}$ and each edge (i, j) translates to user j following user i. The graph G is represented by an adjacency matrix $\mathbf{A}$, of dimensions $N \times N$, where an element A_{ij} equals 1 if user j follows user i. The current paper aims to infer the hidden adjacency matrix $\mathbf{A}$.

Hidden information. Generally, a social media platform provides a Newsfeed and a Wall for each user. The Wall includes the posts and reposts of the users, whereas the Newsfeed includes the posts and reposts created by their respective followees. Newsfeeds are formed based on the friendships in the network. Accordingly, Fig. 2 shows the possible Newsfeeds and Walls of the users in $\mathcal{U}_1$ that created the trace $\mathcal{T}_1$. As we notice, the Newsfeeds are a result of the way that users are connected, i.e., their friendship graph G_1, which is what we are seeking to infer. Walls are filled with individual posts from users and their interaction with Newsfeeds. If we assume that we have access to the unknown Newsfeeds and the corresponding friendship graph

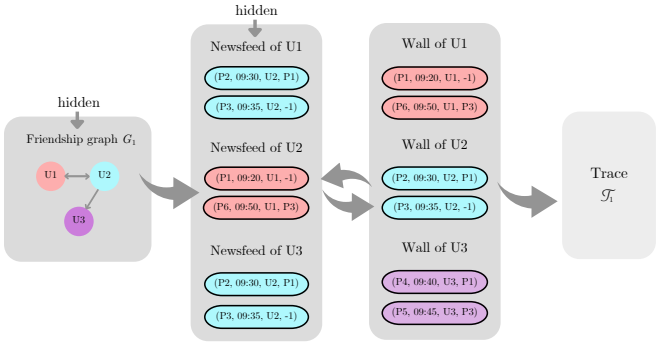


Fig. 2: The hidden way that information diffuses through the ground truth network of users G_1 that produces the trace $\mathcal{T}_1$. Our goal is to infer G_1 (or equivalently its adjacency matrix $\mathbf{A}$ from $\mathcal{T}_1$.

of Fig. 2, we can infer directly how the post P1 observed in $\mathcal{T}_1$ is diffused:

1. It is initially posted by author U1 at t_0=09:20.
2. At timestamp t_0 post P1 appears on the Newsfeed of U1's followers, in this case user U2.
3. At a later timestamp, t_1=09:30, U2 reposts P1 on their Wall. Their repost takes the *pid*=P2.
4. At the same timestamp t_1, P2 appears on the Newsfeed of U2's followers, U1 and U3.
5. Later, at t_2=09:40, U3 reposts P2. Their repost takes the *pid*=P4.

As a result, we inferred that P1 diffused from user U1 to U2 and then to U3 (assuming that users only retweet the users that they follow). Inferring this path was trivial since we assumed that we had access to the Newsfeeds which show the intermediary *pids* of the reposts. However, until today, social media platforms keep Newsfeeds private to each user. Therefore, in the final trace $\mathcal{T}_1$ this information is hidden. Instead, we only have access to the timestamps of the reposts of P1 and the author it is mapped to (user U1). For user U2 it is trivial to infer that they reposted P1 directly from U1 (and thus follow them) since they are the first in the trace to repost it. However, it is non-trivial to infer through whom U3 reposted P1; it could be through any of the users U1 or U2.

Of course, the above example is quite simplistic; we can still come up with some trivial guesses about how the three users are connected that are not very far from the ground truth. In reality, though, we will have to deal with traces that include millions of entries, which makes our task much more challenging. Since social media traces hide the Newsfeeds and the intermediary retweet ids, we do not know the real paths through which posts diffuse: a repost made by each user *uid* points only to the author of the initial post and not to the real user that *uid* reposted. Therefore, due to the trace being only a (partial) view of each user's Wall and their interactions with their (hidden) Newsfeed, we cannot infer the friendship connections between the users directly.

Our intuition is that it is more likely that user j is following user i ($A_{ij} = 1$) if a post reaches often user j through user i (via the edge (i, j)). With this information not being directly available, we aim to infer the intermediary diffusion paths that are hidden in the trace. This will generate the unknown friendship graph G in question. To achieve this, we introduce a set of constraints that guides the graph's inference toward a feasible result.

3.3. Assumptions on the diffusion of posts

To generate the hidden diffusion paths, we first need to decide on a diffusion model. In this case, we opt for a simple model, the SI diffusion model, which has been extensively used in epidemiological models (Daley and Gani, 1999) and apply it to social media users: when a new post arrives on a user's Newsfeed, they are Susceptible to infection. If they choose to repost it they become Infected given the specific post and remain so for the rest of the diffusion. Most existing works using the SI model consider that infection can happen only one time step ahead, after a user becomes Susceptible. We assume, however, that when a user posts a message, they can diffuse it to their still uninfected followers (those in the Susceptible state) during any consecutive timestamp. Furthermore, we make some additional assumptions as follows:

1. The author of every original post that has been reposted is included in the trace $\mathcal{T}$.
2. Users repost *only* from their followees, i.e., the users they follow. We assume that the latter are always present in the available trace.
3. A post can diffuse from user i to user j only if user i has shared the post chronologically earlier in $\mathcal{T}$ than user j.

Although the second assumption does not always hold in practice, it simplifies our task. As we will see later, our approach can be expanded accordingly to take into account instances in which people repost content from followees who are not inside the trace or even from users outside their list of followees (e.g., when Twitter users repost something from the trending hashtags or via the search function, etc). We should also note that we can only obtain friendships between users who have interacted with one another at least once in the available trace $\mathcal{T}$.

3.4. Episodes

We collect the set of all original posts (the ones that have $rid = -1$ in $\mathcal{T}$) that we call $\mathcal{S}$. Each original post $s \in \mathcal{S}$ along with its reposts is called an *episode* and is defined as follows:

Definition 1 (Episode). For each original post $s \in \mathcal{S}$ we define an episode as a set of users $\mathcal{E}_s = author_s \cup \{u \in \mathcal{U} \mid \exists\, (pid, t) : (pid, t, u, s) \in \mathcal{T}\}$. In other words, each episode $\mathcal{E}_s$ includes the author of s, denoted by $author_s$, followed by the users who reposted it, in chronological order. The whole set of episodes is denoted by $\mathcal{E}$ and includes S episodes in total.

To indicate that user i appears in $\mathcal{E}_s$ before j we use the notation $i <^s j$. We call this pair a *temporally ordered pair* $(i, j)_s$. Out of the S total episodes in $\mathcal{T}$, we count M_{ij} where it holds that $i <^s j$. If $M_{ij} > 0$, it is probable that j has reposted content from i. In this case, the pair is referred to as an *active pair*. Our intuition is that we become more certain about the existence of a diffusion path from i to j as M_{ij} becomes larger. As a result, M_{ij} is a quantity that can determine the hidden post-propagation paths and we will use it extensively in the sections that follow. Every piece of information that can be directly derived from a trace $\mathcal{T}$ can be found in Table 1.

Table 1: Information that is directly available from the data.

Symbol	Definition
$\mathcal{T}$	Set of T post instances of the type (pid, t, uid, rid).
$\mathcal{U}$	Set of users that are included in $\mathcal{T}$ ($\|\mathcal{U}\| = N$).
$\mathcal{S}$	Set of original posts in $\mathcal{T}$ ($\|\mathcal{S}\| = S$).
$\mathcal{E}$	Set of episodes in $\mathcal{T}$ ($\|\mathcal{E}\| = S$).
$\mathcal{E}_s \in \mathcal{E}$	Episode of original post s, $1 \leq s \leq S$.
$author_s$	The *uid* of the author of s.
$i <^s j$	User i reposted or posted s before user j.
M_{ij}	# episodes where it holds true that $i <^s j$.

3.5. Feasibility of a trace given an inferred graph

For every episode $\mathcal{E}_s$ in the trace $\mathcal{T}$ and every user i that reposted s before j in time, we define the binary variable $X_{ij}(s) \in \{0, 1\}$ that is equal to 1 if the post s passed from i to j (i.e., j follows i) and 0 otherwise. As underlined in the previous section, the real value of $X_{ij}(s)$ is unknown. Therefore, given the chronological order of reposts in $\mathcal{E}_s$, we may imagine many feasible routes through which the post s might have spread to those who reposted it. These paths create a propagation graph $G_s = \{V_s, E_s\}$ per episode, with the users in each episode $\mathcal{E}_s$ as nodes ($V_s = \mathcal{E}_s$), and the edges set E_s containing the (unknown) edges that we infer for the given post. Every edge that we infer follows the propagation's direction; for instance, an edge (i, j) inferred in G_s indicates that $X_{ij}(s) = 1$. Given the above and our problem definition, for each episode s in $\mathcal{T}$, our goal is to infer a directed acyclic graph (DAG) G_s that is *feasible* and explains the whole $\mathcal{E}_s$ sequence.

Definition 2 (Feasible propagation DAG G_s per episode $\mathcal{E}_s$). Given an episode $\mathcal{E}_s$ from $\mathcal{T}$, we say that a propagation DAG G_s is feasible, or, equivalently, that it explains $\mathcal{E}_s$, if (i) there exists (at least) one directed path from the author $author_s$ to every other user $j \in \mathcal{E}_s \backslash author_s$ and (ii), for each edge (i, j) of the path it holds that $i <^s j$, i.e., all of its edges follow the time-ordering of the reposts. If we take the union of every feasible propagation graph G_s inferred per episode $\mathcal{E}_s$, we get the full friendship graph G and we can build its adjacency matrix $\mathbf{A}$ as follows: we set $A_{ij} = 1$ if there exists at least one G_s where the edge (i, j) exists, and 0 otherwise.

Definition 3 (Feasible friendship graph G). An inferred graph G is called feasible, if, for every episode $\mathcal{E}_s$ in $\mathcal{T}$, there exists a subgraph which is a feasible propagation DAG G_s as we defined it above. Keep in mind that the full graph G is not a DAG.

To make the concept of feasibility more clear, we show in Fig. 3 some examples of possible friendship graphs that could have been inferred, given the example trace $\mathcal{T}_1$. It contains two episodes $\mathcal{E} = \{\mathcal{E}_{S1}, \mathcal{E}_{S2}\}$, where S1 = P1 and S2 = P3. Graph G_A explains episode $\mathcal{E}_{S2}$ by inferring that the post S1 diffused directly from author U2 to users U3 and U1. However, for post S1, there is no feasible path from the author U1, to users U2 and U3 that reposted it. Thus, the episode $\mathcal{E}_{S1}$ is non-feasible given G_A and the final friendship graph G_A is only 50% feasible, since it only explains half the trace. Similarly, graph G_B is only 50% feasible since it does not explain episode $\mathcal{E}_{S2}$: it does not give a feasible propagation path to explain how S2 arrived to U1 from author U2. In contrast, graphs G_C and G_D are both 100% feasible because we can find a feasible propagation graph for each episode in the trace. Therefore, either of the two graphs could be considered a feasible solution to our graph inference problem.

We should note here that there are more combinations of feasible connections that we could think of; these figures demonstrate only two representative feasible examples.

3.6. Inference of post diffusion

3.6.1. Feasibility constraints on reposting behavior

The main challenge of network inference in OSNs arises from the fact that the binary value $X_{ij}(s)$ defined in Section 3.5 for the different user pairs is unknown. However, we can restrict the number of solutions by imposing a set of constraints on all the values. These constraints should ensure that all the episodes in the trace are feasible given the inferred graph according to Definition 1. Specifically, they should guarantee that if a user j appears in an episode $\mathcal{E}_s$ (after the author $author_s$) they should be connected with at least one user i that appears in $\mathcal{E}_s$ before them, including the author of s (i.e., it should hold that $i <^s j$). As a result, the constraints have the following format:

$$\sum_{i \in \mathcal{E}_s \text{ s.t. } i <^s j} X_{ij}(s) \geq 1, \forall j \in \mathcal{E}_s \backslash \{author_s\}, \tag{1}$$

$$X_{ij}(s) \in \{0, 1\}, \ \forall i, j \in \mathcal{U}, \ \forall s \in \mathcal{S}. \tag{2}$$

Fig. 3 shows the constraints on $X_{ij}(s)$, given the set of episodes $\mathcal{E} = \{\mathcal{E}_{S1}, \mathcal{E}_{S2}\}$. The role of these constraints is to guide the process toward solutions that belong to the feasible group of graphs. To do so, the constraints should be defined for each episode $\mathcal{E}_s \in \mathcal{E}$, and each user that reposted s, according to Eq. 1. For example, as we see in Fig. 3, given the first constraint for episode $\mathcal{E}_{S1}$, we can derive easily that the user U2 reposted post S1 directly from its author U1 ($X_{12}(\text{S1}) = 1$). The second constraint tells us that user U3 has reposted S1 either from U1, or from U2 (or, from both). If we look closer, the possible graphs that we marked as non-feasible earlier violate these constraints. For example, G_A violates the first constraint for $\mathcal{E}_{S1}$, since $X_{12}(\text{S1}) = 0$. Likewise, G_B violates the last constraint for $\mathcal{E}_{S1}$, since $X_{21}(\text{S2}) + X_{31}(\text{S2}) = 0$. As we saw in the figure and the equations above, the X_{ij} value of a pair (i, j) is different for each episode that it appears in. For example, X_{23} appeared two times, one time for $S1$ and another one for S2.

With all the possible combinations that each X_{ij} value can take for all active pairs and episodes observed, we soon realize that the problem is intractable when dealing with large traces. The only direct knowledge we have for each pair is the constant value M_{ij}, i.e., the total number of times that a user i appears before j in every episode $\mathcal{E}_s \in \mathcal{T}$. What we are interested in, is the number of times that a post diffused through the edge (i, j), out of the M_{ij} times that it could be possible. We model this with the unknown quantity Y_{ij} which is equal to the total number of times that j reposts from i. More formally:

$$Y_{ij} = \sum_{s \in \mathcal{S}, \text{ s.t. } i <^s j}^{S} X_{ij}(s). \tag{3}$$

As we can see above, to find Y_{ij}, we sum over all episodes where it holds that $i <^s j$. This happens M_{ij} times in total.

Diffusion probabilities. To solve the problem we make the following important assumption: for every active pair (i, j) in any episode $\mathcal{E}_s \in \mathcal{E}$, a user j reposts an s from i independently from other episodes with an unknown diffusion probability $\sigma_{ij} \in [0, 1]$. Therefore, $X_{ij}(s)$ is an independent Bernoulli random variable with a mean parameter σ_{ij} which does not depend on s. In other words, the diffusion probability σ_{ij} of a user

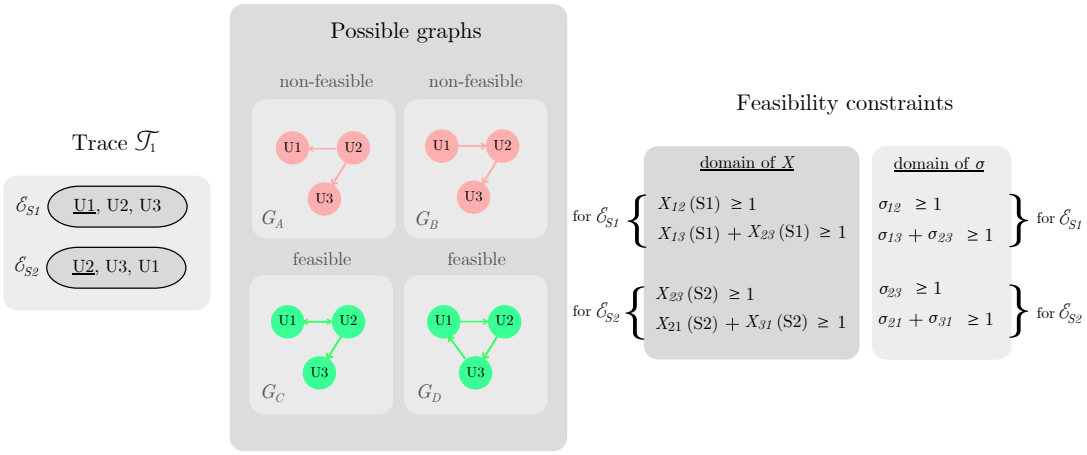


Fig. 3: Feasibility check of different inferred graphs given a trace $\mathcal{E} = \{\mathcal{E}_{S1}, \mathcal{E}_{S2}\}$.

pair is the same across all episodes, which means that there is no preference in terms of content when someone chooses to repost. Of course, this does not accurately reflect reality but it serves as a useful simplification. Therefore, for an ordered user pair $(i, j)_s$, σ_{ij} equals:

$$\sigma_{ij} = \mathbb{E}\left[X_{ij}(s)\right]. \tag{4}$$

We can now transfer our problem from searching over the binary domain of $X_{ij}(s)$ to solving over the real domain of the σ_{ij} values. By taking the expectation in (1) and given Eq. 4, we get the following set of constraints:

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

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

From Eq. 3 and Eq. 4, Y_{ij} is the sum of M_{ij} independent Bernoulli random variables that have a mean value σ_{ij}. In other words, Y_{ij} is an independent Binomial random variable with mean value $M_{ij}\sigma_{ij}$:

$$\mathbb{E}[Y_{ij}] = \sum_{s \in \mathcal{S}, \text{ s.t. } i <^s j}^{\mathcal{S}} \mathbb{E}[X_{ij}(s)] = \sum_{s \in \mathcal{S}, \text{ s.t. } i <^s j}^{\mathcal{S}} \sigma_{ij} = M_{ij}\sigma_{ij}. \tag{7}$$

4. Problem Modeling and Learning Method

We introduce a feasible inference method, called CEM-*, with two special cases, depending on the assumed distribution of the underlying graph. The first case assumes an Erdős–Rényi (ER) prior and is called CEM-er. According to this prior, the underlying graph that we are trying to infer has been created under a uniform probability ρ that is the same for all edges. However, this does not accurately reflect the structure of social media graphs, which are less random and have some important properties, such as the existence of hubs. After this section, we propose an additional case that incorporates a more realistic model for the underlying graph, the stochastic block model (SBM). We call this extended method CEM-sbm.

4.1. Erdős–Rényi prior (CEM-er)

As mentioned above, the prior structure of the network $\mathbf{A}$ is not known, and therefore a uniform prior ρ is assumed for all edges. Hence, the prior takes the form of a probability distribution $P(\mathbf{A} \mid \theta)$, where θ is a set of hidden parameters that give us more details on the underlying network. Given a trace $\mathcal{T}$ of posts and reposts, $P(\mathbf{A}, \theta \mid \mathcal{T})$ is the probability that the inferred graph is $\mathbf{A}$ and the parameters get the value θ. The parameters θ should account for a wider range of potential graph types and data generation methods. Therefore, they are chosen as follows:

- The probability that a user j shares content through a user i, represented by the set of σ_{ij} values that we presented in Section 3.6.1.
- To model the uncertainty about the structure of the graph's adjacency matrix $\mathbf{A}$, we assumed that there is a prior probability ρ of an edge drawn independently between any two nodes i, j (Erdős–Rényi prior).
- The *true positive utilization rate* α: the probability of a post propagating through an edge that we inferred to exist in the underlying graph G. Given the (hidden) number of interactions between users Y_{ij}, we consider that when an edge exists in G ($A_{ij} = 1$) the Y_{ij} out of the M_{ij} experiments are successful (we get Y_{ij} true positive edges in total), each with probability α.
- The *false positive utilization rate* β: the probability of inferring that a post propagated through edges that do not exist in G. Likewise to above, when $A_{ij} = 0$, we consider that the Y_{ij} out of M_{ij} experiments are successful (we get Y_{ij} false positive edges), each with probability β.

We can see that the global parameters α and β depend on whether an edge exists in the ground truth graph G. To find the most probable value of the parameters θ given the observed data and infer $\mathbf{A}$ with maximum likelihood, we will employ an Expectation–Maximization (EM) algorithm which is a standard inference tool when some data is unknown or hidden. As suggested by its name, an EM iteration involves two consecutive steps: an expectation (E) step, which computes the expected log-likelihood under the most recent estimation of the parameters in θ; then, a maximization (M) step, which determines the parameters that maximize the expectation. Then, the computed parameters are used in the following iteration, and so on, until we satisfy a convergence criterion.

We start constructing the EM iterations, following the method proposed by Newman (2018) and employ the Bayes' theorem:

$$P(\mathbf{A}, \theta \mid \mathcal{T}) = \frac{P(\mathcal{T} \mid \mathbf{A}, \theta) P(\mathbf{A} \mid \theta) P(\theta)}{P(\mathcal{T})}. \tag{8}$$

The probability that we get the specific set of posts and reposts $\mathcal{T}$ given $\mathbf{A}$ and the parameters $\theta = \{\alpha, \beta, \rho, \boldsymbol{\sigma}\}$, found in the numerator of the above expression, will differ 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. We also assumed as prior knowledge that between any two nodes in $\mathbf{A}$ an edge has been drawn with probability ρ. Therefore we get:

$$P(\mathcal{T} \mid \mathbf{A}, \theta) P(\mathbf{A} \mid \theta) = \prod_{i \neq j} \left[\alpha^{Y_{ij}} (1-\alpha)^{M_{ij} - Y_{ij}} \rho\right]^{A_{ij}} \left[\beta^{Y_{ij}} (1-\beta)^{M_{ij} - Y_{ij}} (1-\rho)\right]^{1 - A_{ij}}. \tag{9}$$

Given this type of model, 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 (8) over all possible networks $\mathbf{A}$, we find that $P(\theta \mid \mathcal{T}) = \sum_{\mathbf{A}} P(\mathbf{A}, \theta \mid \mathcal{T})$. Then, as suggested by Newman (2018), we can apply the well-known Jensen's inequality on the log of $P(\theta \mid \mathcal{T})$:

$$\log P(\theta \mid \mathcal{T}) = \log \sum_{\mathbf{A}} P(\mathbf{A}, \theta \mid \mathcal{T}) \geq \sum_{\mathbf{A}} q(\mathbf{A}) \log \frac{P(\mathbf{A}, \theta \mid \mathcal{T})}{q(\mathbf{A})}, \tag{10}$$

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 $Q_{ij} = P(A_{ij} = 1 | \mathcal{T}, \theta) = \sum_{\mathbf{A}} q(\mathbf{A}) A_{ij}$. If we take the expectation of Eq. (10) we find that:

$$\mathbb{E}[\log P(\theta \mid \mathcal{T})] \geq \sum_{\mathbf{A}} q(\mathbf{A}) \log \frac{D_{ij}}{q(\mathbf{A})}, \tag{11}$$

$$\text{where } D_{ij} = \Gamma \prod_{i \neq j} \left[\rho \alpha^{M_{ij}\sigma_{ij}} (1-\alpha)^{M_{ij}(1-\sigma_{ij})}\right]^{A_{ij}} \left[(1-\rho) \beta^{M_{ij}\sigma_{ij}} (1-\beta)^{M_{ij}(1-\sigma_{ij})}\right]^{1-A_{ij}}. \tag{12}$$

We find that the choice of q that achieves equality of (11) and hence, maximizes the right-hand side with respect to q is:

$$q(\mathbf{A}) = \prod_{i \neq j} Q_{ij}^{A_{ij}} (1 - Q_{ij})^{1 - A_{ij}}, \tag{13}$$

where, Q_{ij} is the posterior probability that the edge (i, j) exists, and we find that it equals:

$$Q_{ij} = \frac{\rho \alpha^{M_{ij}\sigma_{ij}} (1-\alpha)^{M_{ij}(1-\sigma_{ij})}}{\rho \alpha^{M_{ij}\sigma_{ij}} (1-\alpha)^{M_{ij}(1-\sigma_{ij})} + (1-\rho) \beta^{M_{ij}\sigma_{ij}} (1-\beta)^{M_{ij}(1-\sigma_{ij})}}. \tag{14}$$

The details of the above derivation are shown in Appendix A. Hence, to find the maximizing posterior distribution $q(\mathbf{A})$ it suffices to find the individual maximizing posterior probabilities Q_{ij} according to Eq. (14). Given these values, if we further maximize with respect to the parameters $\theta = \{\alpha, \beta, \rho, \boldsymbol{\sigma}\}$ we can get the maximum-likelihood value we seek. The updates for the first three parameters are thus calculated to be the following:

$$\alpha = \frac{\sum_{i \neq j} M_{ij} \sigma_{ij} Q_{ij}}{\sum_{i \neq j} M_{ij} Q_{ij}}, \; \beta = \frac{\sum_{i \neq j} M_{ij} \sigma_{ij} (1 - Q_{ij})}{\sum_{i \neq j} M_{ij} (1 - Q_{ij})}, \tag{15}$$

$$\rho = \frac{1}{N(N-1)} \sum_{i \neq j} Q_{ij}, \tag{16}$$

where N is the number of users in the trace. Finally, to find the whole vector $\boldsymbol{\sigma}$ that includes all the σ_{ij} unknown diffusion parameters, we must solve a linear optimization problem as follows (for derivation refer to Appendix A):

$$\max_{\sigma} \sum_{i \neq j} \sigma_{ij} (W_{ij} - \lambda c) \tag{17}$$

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

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

$\lambda > 0$ some given penalty for regularisation, and $c = \max_{(i,j) \in W} W_{ij}$.

We added the value λ into the optimization objective as a penalty per iteration, since our initial goal is to infer a graph that is feasible with the minimum possible number of edges. 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 we initially wanted. As λ moves closer to 1, it 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. As λ approaches 0, the optimization infers the largest number of edges possible. We will explore in detail the effect of the hyperparameter λ with values that vary from 0 to 1 in the Experiments section. The final CEM-er algorithm is shown in Algorithm 1.

4.2. Stochastic block model prior (CEM-sbm)

Since we are working with social media data, where there is usually a strong presence of communities, we believe it is more realistic to assume that the network is derived from a stochastic block model (SBM), a generative model of community structure that was first proposed in the 1980s by Holland et al. (1983). In the standard SBM, each node i participates in a different block (community) which we indicate by g_i, where i may take values in $[1, G]$ where G is the number of hidden communities. The number of edges between nodes i and j follows a Bernoulli distribution with mean ω_{g_i, g_j}, that is the relative probability of intra-community (if $g_i = g_j$) or inter-community (if $g_i \neq g_j$) connection.

As we can see, in the case of CEM-er, the prior structure of the network $\mathbf{A}$ was the only kind of unobserved data, but in this case, we have two unknowns: the network $\mathbf{A}$ and the vector of the group assignments of the users $\mathbf{g}$. Hence, the prior takes the form of a probability distribution $P(\mathbf{A}, \mathbf{g} \mid \theta)$, where θ denotes the unknown parameters of the distribution, which gives additionally the details of the community structure. This approach, therefore, allows us to infer both the unknown network structure and the community structure simultaneously. Given a trace $\mathcal{T}$, $P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{T})$ is the probability that we get $\mathbf{A}$, the users' community participation vector $\mathbf{g}$ and a set of chosen parameters θ. The parameters set θ that we select here includes two newly added parameters that replace the prior ρ that we had in the CEM-er case:

– Following the SBM for $\mathbf{A}$ and the users' community participation vector $\mathbf{g}$, we suppose that there is a prior probability p of an edge existing between any two nodes i, j that belong in the same community, i.e., $g_i = g_j$.

– The nodes that belong in different communities are connected with a probability q.

We construct the EM iterations as we did before, following Bayes' theorem:

$$P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{T}) = \frac{P(\mathcal{T} \mid \mathbf{A}, \mathbf{g}, \theta) P(\mathbf{A}, \mathbf{g} \mid \theta) P(\theta)}{P(\mathcal{T})}. \tag{18}$$

Taking into consideration the definition of the parameters above, the probability that we get the specific trace $\mathcal{T}$, given $\mathbf{A}$, $\mathbf{g}$ and $\theta = \{\alpha, \beta, p, q, \boldsymbol{\sigma}\}$ is driven by the probabilities α and β, whereas the probability that we get $\mathbf{A}$ and $\mathbf{g}$ given θ depends on the probabilities p and q. Therefore, assuming that each user reposts independently from others:

$$\begin{aligned} P(\mathcal{T} \mid \mathbf{A}, \mathbf{g}, \theta) P(\mathbf{A}, \mathbf{g} \mid \theta) = & \prod_{\substack{i \neq j \\ g_i = g_j}} \left[\alpha^{Y_{ij}} (1-\alpha)^{M_{ij} - Y_{ij}} p\right]^{A_{ij}} \\ & \left[\beta^{Y_{ij}} (1-\beta)^{M_{ij} - Y_{ij}} (1-p)\right]^{1 - A_{ij}} \prod_{\substack{i \neq j \\ g_i \neq g_j}} \left[\alpha^{Y_{ij}} (1-\alpha)^{M_{ij} - Y_{ij}} q\right]^{A_{ij}} \\ & \left[\beta^{Y_{ij}} (1-\beta)^{M_{ij} - Y_{ij}} (1-q)\right]^{1 - A_{ij}}. \end{aligned} \tag{19}$$

For the whole set of parameters θ, we assume again a uniform prior probability $P(\theta)$. We sum (18) over all possible networks $\mathbf{A}$ and we find that $P(\theta \mid \mathcal{T}) = \sum_{\mathbf{A}} P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{T})$. Then, we can apply the well-known Jensen's inequality on the log of $P(\theta \mid \mathcal{T})$:

$$\log P(\theta \mid \mathcal{T}) = \log \sum_{\mathbf{A}} P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{T}) \geq \sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) \log \frac{P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{T})}{q(\mathbf{A}, \mathbf{g})}, \tag{20}$$

where $q(\mathbf{A}, \mathbf{g})$ is any joint probability distribution over networks $\mathbf{A}$ and group assignments $\mathbf{g}$ satisfying $\sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) = 1$. We also define the posterior probability of an edge existing between i and j that belong to communities g_i, g_j by $Q_{ij}(g_i, g_j) = P(A_{ij} = 1 \mid \mathcal{T}, \theta) = \sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) A_{ij}$.

For the E-step of the EM algorithm, following the same derivation logic as in the CEM-er variation (in detail in Appendix B), we find that $Q_{ij}(g_i, g_j)$ is the posterior probability that the edge (i, j) exists and is different depending on whether users i, j belong in the same community($g_i = g_j = r$) or not ($g_i = r, g_j = s$, $r \neq s$) with $r, s \in g$:

$$Q_{ij}(r, r) = \frac{p \alpha^{M_{ij}\sigma_{ij}} (1-\alpha)^{M_{ij}(1-\sigma_{ij})}}{p \alpha^{M_{ij}\sigma_{ij}} (1-\alpha)^{M_{ij}(1-\sigma_{ij})} + (1-p) \beta^{M_{ij}\sigma_{ij}} (1-\beta)^{M_{ij}(1-\sigma_{ij})}}, \tag{21}$$

$$Q_{ij}(r, s) = \frac{q \alpha^{M_{ij}\sigma_{ij}} (1-\alpha)^{M_{ij}(1-\sigma_{ij})}}{q \alpha^{M_{ij}\sigma_{ij}} (1-\alpha)^{M_{ij}(1-\sigma_{ij})} + (1-q) \beta^{M_{ij}\sigma_{ij}} (1-\beta)^{M_{ij}(1-\sigma_{ij})}}. \tag{22}$$

Notice that for $M_{ij} = 0$, $Q_{ij}(g_i, g_j)$ becomes equal to the prior probability p if $g_i = g_j$ and equal to q if $g_i \neq g_j$. Next, to maximize the likelihood in terms of the parameters we find:

$$\alpha = \frac{\sum_{i \neq j} M_{ij} \sigma_{ij} Q_{ij}(g_i, g_j)}{\sum_{i \neq j} M_{ij} Q_{ij}(g_i, g_j)}, \tag{23}$$

$$\beta = \frac{\sum_{i \neq j} M_{ij} \sigma_{ij} (1 - Q_{ij}(g_i, g_j))}{\sum_{i \neq j} M_{ij} (1 - Q_{ij}(g_i, g_j))}, \tag{24}$$

$$p = \frac{1}{\sum_{i \neq j} \mathbf{1}(g_i = g_j)} \sum_{i \neq j, g_i = g_j} Q_{ij}(g_i, g_j), \tag{25}$$

$$q = \frac{1}{\sum_{i \neq j} \mathbf{1}(g_i \neq g_j)} \sum_{i \neq j, g_i \neq g_j} Q_{ij}(g_i, g_j). \tag{26}$$

To find the diffusion probabilities σ_{ij} we must solve the following linear optimization problem:

$$\max_{\boldsymbol{\sigma}} \sum_{i \neq j} \sigma_{ij} (W_{ij} - \lambda c) \tag{27}$$

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

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

$\lambda > 0$ some given penalty for regularisation, and $c = \max_{(i,j) \in W} W_{ij}$.

The final CEM-* algorithm, when we choose the SBM prior is shown in Algorithm 1. It iterates between finding an optimal value for q, via the Q_{ij} values, and then holding it constant to maximize the likelihood (the right-hand side of (B.3) in Appendix B) with respect to $\theta = \{\alpha, \beta, p, q, \boldsymbol{\sigma}\}$ (M-step). We underline that the updates of the Q_{ij} values that are essential for the E-step require the knowledge of the communities participation vector $\mathbf{g}$. It is updated in each iteration as follows: we generate first a graph from the current Q_{ij} estimations, by drawing an edge whenever $Q_{ij} > 0.5$. To get the updated vector $\mathbf{g}$, we apply to the generated graph the Louvain method, which returns a single community label for each user node (Blondel et al. 2008). Our algorithm converges when the L2 norm of improvement $\|\mathbf{Q}_{new} - \mathbf{Q}_{old}\|$ falls under some threshold ϵ that we choose in advance, where $\mathbf{Q}$ is the matrix with the Q_{ij} values.

5. Methodology

5.1. Datasets

The general framework of CEM-* for both priors is shown in Fig. 4. To evaluate our two methods CEM-er and CEM-sbm against the ground truth and compare them with existing methods, we will use two different datasets: a synthetic and a real-world one. The synthetic dataset that we create aims to illustrate our method's efficiency when the trace includes sufficient information about the interactions between users. As we will show later, this is not always the case with real-world traces coming from OSNs, which can make the inference task even more challenging.

5.1.1. Synthetic dataset

For the generation of synthetic social media data, we follow the code found in Giovanidis et al. (2021). We first create a set of 100 users each of which has two random activity (posting and reposting) rates. Then, we create an SBM graph between the users, with 7 different partitions of varying sizes, that represents the friendship graph of the network. Users in the same group are connected with probability $p = 0.06$ and users of different groups are connected with probability $q = 0.007$. Each subgraph corresponding to a group is a random Erdős–Rényi with connection probability p. For each user, we generate a set of random timestamps, that increase according to an exponential distribution that depends on their activity rates. These timestamps represent the times they posted or reposted something. We generate a set of 100,000 timestamp-user-activity instances in total that we call the Events set.

Additionally, we assume that each user has a Newsfeed that can hold up to 10 posts and reposts from their followees. Based on the

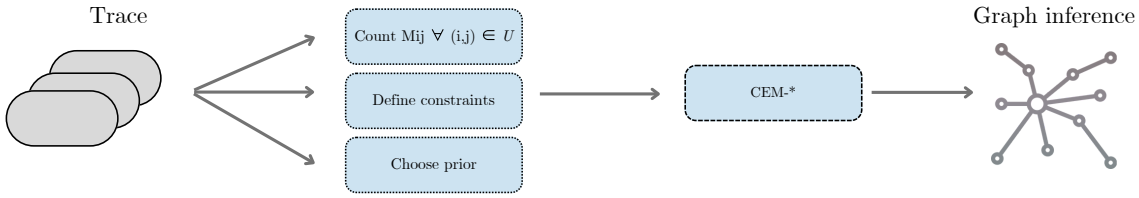


Fig. 4: Framework of Constrained-EM

Table 2: Trace and ground truth statistics for the synthetic and real-world data.

Trace statistics	Synthetic	#Élysée2017fr
Time-span	17,459 time-steps	6 months
# tweets	1,709	293,405
# retweets	24,347	1,605,059
# users	100	11,521
% users with # tweets > 0	27.00	70.74
% users with # retweets > 0	87.00	96.45
% user pairs with $M_{ij} > 0$	78.10	5.21

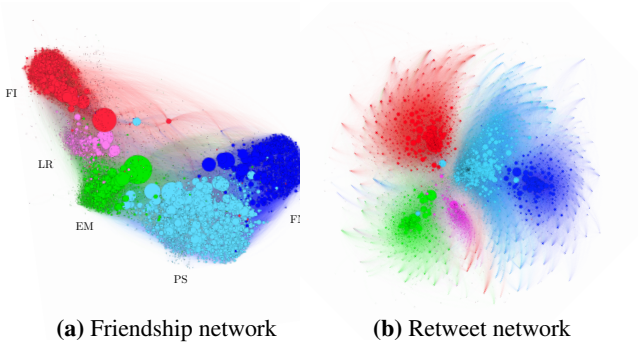
Ground-truth graph	Synthetic	#Élysée2017fr
# edges	158	1,555,718
% intra-edges(labeled)	63.92 (101)	84.29 (1,311,463)
% inter-edges(labeled)	36.08 (57)	15.71 (244,255)
% edges with $M_{ij} > 0$	99.36 (155)	45.23 (703,682)
% intra-edges with $M_{ij} > 0$	100.00 (101)	50.13 (657,389)
% inter-edges with $M_{ij} > 0$	98.25 (56)	18.95 (46,293)

Algorithm 1 CEM-*

Input: \$PRIOR, C, $\mathcal{U}$, M_{ij} $\forall(i,j) \in \mathcal{U}$
Output: $Q, \alpha, \beta, \rho, \boldsymbol{\sigma}$
t = 0
Random Initialisation: $\alpha_t, \beta_t, \rho_t, \boldsymbol{\sigma}_t$
if \$PRIOR = ER **then**
  repeat
    t += 1
    Q_t = update $Q(\alpha_{t-1}, \beta_{t-1}, \rho_{t-1}, \boldsymbol{\sigma}_{t-1})$ using (14)
    α_t = update $\alpha(Q_t, \boldsymbol{\sigma}_{t-1})$ using (15)
    β_t = update $\beta(Q_t, \boldsymbol{\sigma}_{t-1})$ using (15)
    ρ_t = update $\rho(Q_t)$ using (16)
    $\boldsymbol{\sigma}_t$ = update $\boldsymbol{\sigma}(Q_t, \alpha_{t-1}, \beta_{t-1})$ using (17)
  until convergence
else if \$PRIOR = SBM **then**
  Random Initialisation: $\mathbf{g}_t$
  repeat
    t += 1
    Q_t = update $Q(\alpha_{t-1}, \beta_{t-1}, p_{t-1}, q_{t-1}, \boldsymbol{\sigma}_{t-1}, \mathbf{g}_{t-1})$ using (21), (22)
    α_t = update $\alpha(Q_t, \boldsymbol{\sigma}_{t-1})$ using (23)
    β_t = update $\beta(Q_t, \boldsymbol{\sigma}_{t-1})$ using (24)
    p_t = update $\rho(Q_t)$ using (25)
    q_t = update $\rho(Q_t)$ using (26)
    $\boldsymbol{\sigma}_t$ = update $\boldsymbol{\sigma}(Q_t, \alpha_{t-1}, \beta_{t-1})$ using (27)
    $\mathbf{g}_t$ = LOUVAIN(Q_t)
  until convergence
end if

friendship graph and the Events set, we simulate a set of interactions between the users according to the following scheme: when a user i visits their Newsfeed, they repost randomly one of the 10 entries made by their followees. A new entry on the Newsfeed list will push out an older entry of a random position. The Newsfeeds of the users that follow user i will then be updated accordingly. Of course, in reality, users on a social media platform may show a preference towards a specific account or topic, or even repost something outside of the scope of their followees. The random uniform selection, however, makes the simulation collect sufficient information for all the edges in the friendship graph.

The simulation generates a social media trace from which we can extract all the quantities that are necessary for our method, as presented in Section 3. The detailed statistics of the synthetic dataset can be found



(a) Friendship network **(b)** Retweet network

Fig. 5: Networks of the 5 political parties provided by the #Élysée2017fr dataset.

in Table 2. The table on the left shows the statistics of the trace, whereas the table on the right shows the statistics of the ground truth graph. This is the graph that we will be trying to infer. The intra-edges refer to the edges inside a community, whereas inter-edges refer to the edges between different communities.

5.1.2. Real-world data: the #Élysée2017fr dataset

For the evaluation of our method on real-world data, we choose #Élysée2017fr, a publicly available dataset related to the 2017 French presidential campaign on Twitter (Fraisier et al. 2018). It features 2,414,584 tweets and 7,763,931 retweets from 22,853 Twitter profiles discussing the election. Users have been manually annotated by experts with political affiliations expressing support for one of the 5 main competing parties in France:

- FI: France Insoumise, far-left (Jean-Luc Mélenchon)
- PS: Parti Socialiste, left-wing (Benoît Hamon)
- EM: En Marche, center (Emmanuel Macron)
- LR: Les Républicains, right-wing (François Fillon)
- FN: Front National, far-right (Marine Le Pen)

The exact timestamps of interactions between users had not been published by the creators of the dataset, therefore we had to crawl them using the Twitter API. On top of that, we collected the follower-followee connections, i.e., the friendship graph of the observed user ids, which had not been provided by the authors. A visual representation of the friendship network that we scraped along with the community participation of each node is shown in Fig. 5a. The network of retweets is

shown in Fig. 5b. From these figures, we can see that even though users follow people from other communities (e.g., there are many friendships between the two extreme groups FI and FN), they mostly retweet posts from authors that belong inside their community and they do not interact much with users outside.

From this trace, we keep only the tweets that have been retweeted by at least one user. Additionally, we remove retweets for which we do not know the author and retweets that have been made more than once by the same user. The statistics of the trace after the above preprocessing along with the statistics of the ground truth graph are shown in Table 2.

Insufficient information in a real-world trace. From Table 2 and Fig. 5a and 5b, we notice the main challenge in working with this dataset against the synthetic one: out of the 1,555,718 edges in the underlying friendship graph, only 45.23% of them have a non-zero M_{ij} value. On the other hand, the synthetic trace includes information for more than 99% of the 158 existing edges. This can be partly because, in reality, users may repost their followees with some preference, instead of randomly selecting posts from their Newsfeed as is the case in the synthetic dataset. Therefore, many users may not appear to interact with retweets even if there is a connection between them in reality. However, given that the absolute numbers of the real-world trace are quite high, we believe that there is sufficient information to work with.

5.2. Comparison

5.2.1. Compared models

We compare the graphs inferred by our two models, CEM-er and CEM-sbm, with those generated by the following baseline and state-of-the-art methods:

- **Star**: a heuristic graph inference method that draws a directed edge from the author of every tweet s in the trace to every user that appears in the corresponding episode $\mathcal{E}_s$ after them. The graph inferred by Star implies that all the users that have retweeted a tweet are following its author.
- **Chain**: another heuristic method that generates a single long path between the users in each episode $\mathcal{E}_s$, according to the timestamps of their interactions with tweet s: each path first connects the author of s to the user i that retweeted it first in time. Then, it connects i to the user j who retweeted it second in time, j to the user who retweeted it third, and so on.
- **Saito et al.** (2008): a baseline EM-based algorithm that infers the influence probabilities k_{ij} by assuming an Independent Cascade model of diffusion between the users. For comparison, we produce the final graph by drawing an edge (i, j) whenever $k_{ij} > 0.5$.
- **Netinf** (2012): in a similar way to our work, Gomez-Rodriguez et al. identify the graph that most accurately explains the observed infection times of nodes. However, their formulation of the problem is combinatorial and thus NP-hard to solve exactly. Therefore, they suggest finding near-optimal networks using approximation algorithms, by exploiting the submodularity properties of the objective, which, as we will show in the next sections, introduces computation-time and precision issues. In contrast, we devise a continuous linear expression based on the trace, which allows us to find efficiently the exact solution to an LP optimization problem.
  As explained by the authors, when the activity rates are not the same for all users, the performance of the model worsens. Therefore, we expect Netinf to perform worse than CEM-* in more realistic settings such as these of the synthetic dataset, in which users have different activity rates. It should be noted that Netinf requires that we set in advance the parameter k, which is the number of edges that we want to infer. For comparison, we set k equal to the number of edges of the corresponding ground truth graph.
- **Newman** (2018): a more recent EM-based algorithm that we introduced in Section 1. As mentioned before, our algorithm is an extension of the EM formulation provided in Newman's work. The algorithm is not designed to consider hidden paths between users, thus it is not guaranteed that the inferred networks will be feasible. For evaluation, we derive a graph by drawing an edge (i, j) whenever the friendship probability Q_{ij} for a user pair (i, j) estimated by this method is greater than 0.5.
- **Peixoto** (2019): a state-of-the-art non-parametric Bayesian method that infers posterior distributions from trace observations using a stochastic block model as a prior. As is the case with our CEM-sbm model, it performs community detection together with network reconstruction. Unlike us, however, during the inference process, the model performs sampling using a Markov Chain Monte Carlo procedure and accepts a solution with a Metropolis-Hastings probability. As demonstrated next, this negatively impacts the computation time of the optimization.

5.2.2. Comparison metrics

The directed edges inferred by each inference method translate to the existence of follower-followee relationships between the respective user nodes. To evaluate and compare them against the ground truth, we will look at the following aspects:

1. **Results of CEM-* given different trace sizes and values of hyperparameter λ.** Firstly, we check how different trace sizes change the corresponding results of our method. For example, by choosing only the first 10,000 lines of the synthetic trace, we obtain information for around 65% out of the $N(N-1) = 9,900$ possible user pairs, whereas the whole trace (= 100,000 lines) informs us on about 78% of the pairs. We see therefore that as we choose more trace lines from the input, we get more information between users in terms of tweets and retweets (with diminishing returns). In general, we expect the performance of our model to improve with the increasing size of the trace.
2. **Feasibility of the trace.** We evaluate each method presented in Section 5.2.1 in terms of feasibility. Given the ground truth graph, we check how many episodes are feasible, according to our definition of feasibility provided in Section 3.5.
3. **Prediction performance.** When the ground truth is available, we can treat the output of the inference as a binary classification task between existing and non-existing edges. We, therefore, choose Precision, Recall, and AUC scores as metrics for evaluation and comparison. These metrics are used frequently to measure prediction success in similar classification tasks. Precision refers to the percentage of true positive friendships inferred out of all the predicted ones, and Recall quantifies the percentage of true positive friendship edges inferred out of all the edges that are positive in the ground truth. The AUC score is the area under the ROC curve that represents the tradeoff between Recall (true positive rate) and Specificity (false positive rate), not to be mixed with the true and false positive utilization rates α and β in the parameters set θ of CEM-*. It is a measure of separability and quantifies how well the model can distinguish between classes.
4. **Inferred network metrics.** Additionally, we look into different network measures of the inferred graph (e.g., average degree, diameter, connected components, etc), and compare them to these of the ground truth graph. These measures can be indicative of how much the inferred graph resembles the properties of a general real-world graph (in cases when the ground truth is not available).
5. **Detection of communities.** A useful by-product of our CEM-sbm network reconstruction method is the community detection task. Therefore, we check to what extent the inferred communities resemble the real ones presented in the ground truth. Since a node can only belong to one community, we wish to verify whether the different pairs of users belonging to the same or different communities are the same in the ground truth. The method for the evaluation and comparison is the following: we first generate a graph for each model as described in Section 5.2.1 and then apply on it the Louvain method for community detection (Blondel et al. 2008). The detected clusters are then used to calculate the F1-score as follows: we look at each possible user pair and if the users belong to the same community we label the edge with 1 (positive class), otherwise with 0 (negative class). We do the same for the ground

truth (with the Louvain labels). From the true/false positive, and true/false negative rates we measure the F1-score, which combines Precision and Recall. In addition, we estimate the values of p and q between the communities in the inferred graph and compare them to the real ones.

5.3. Experimental settings

We run the experiments on a virtual machine with 40 vCPUs and 256 GB RAM. For the solution to the optimization problem, we configure a Gurobi solver through PuLP[3], an open-source linear programming library for Python, using the dual simplex optimization method. The parameters set $\theta_1 = \{\alpha, \beta, r, \sigma\}$ and $\theta_2 = \{\alpha, \beta, p, q, \sigma\}$ for CEM-er and CEM-sbm respectively are initialized uniformly at random in the range $[0, 1]$. As a convergence criterion for the optimization we choose the L2 norm of the difference between the values of $\mathbf{Q}$, i.e., $\|\mathbf{Q}_{new} - \mathbf{Q}_{old}\| < \epsilon$, where the threshold ϵ is set equal to 0.001. Finally, to generate the unknown friendship graph G, we round up all edges with $Q_{ij} > 0.5$ to 1, and the rest are set to 0. We run the experiments 10 times and report the average results.

6. Experiments on synthetic data

Table 3: Converged parameters for $|T_{synth}| = 50{,}000$ lines.

CEM-* parameters	1-α^*	β^*
CEM-er ($\lambda = 0$)	1-(7e-11)	1.74e-12
CEM-er ($\lambda = 1$)	1-(2e-10)	1.54e-13
CEM-sbm ($\lambda = 0$)	1-(7e-11)	1.65e-12
CEM-sbm ($\lambda = 1$)	1-(9e-11)	3.95e-15

6.1. Results of our method (CEM)

Values of parameters. The converged parameters of both our methods, CEM-er and CEM-sbm are shown in Table 3. In the first column, we show $(1 - \alpha^*)$ to be precise about how small the distance is from the maximum value of α^* that is equal to 1. We observe that in every case α is close to 1. This means that there is an almost 100% probability that a post propagated through an edge present in the network we inferred. On the other hand, the small values of β suggest that the number of false positive utilized edges is close to zero. This suggests that a post from the trace always propagates through an edge that has been inferred.

Different sizes of input. Figure 6a shows the relation of Precision and Recall given trace sizes that range from 10,000 to 100,000 lines. As we observe, the larger the trace, the higher the value of Recall. This was expected since bigger traces give more information which helps us derive more underlying edges. Precision presents relatively stable behavior and is higher (=0.869) when $\lambda = 1$. Overall, we see that CEM-sbm has higher performance than CEM-er in terms of Precision which reaches up to 0.869 when $\lambda = 1$, and a slightly worse, but still competitive performance in terms of Recall (reaching up to 0.944 for $\lambda = 1$ whereas CEM-er can reach up to 0.954 for $\lambda = 0$). We conclude therefore that CEM-sbm is much more precise than CEM-er in the case of the synthetic dataset and can also retrieve most of the underlying edges.

Different values of the hyperparameter λ. In Fig. 6b we can see more clearly how the choice of the hyperparameter λ inside the optimization objective (Eq. 17 and Eq. 27) affects the precision of inference: for $\lambda = 0$ we get very low Precision (= 0.024) regardless of the prior since we infer the largest number of edges possible according to the objective, which in turn results to more false positive edges. However, in this case, the Recall value is at its highest (for example 0.954 in the case of CEM-er). In contrast, for $\lambda = 1$, we infer a graph with the smallest number of edges possible given the constraints and thus we get a considerably better Precision (= 0.869, CEM-sbm). The Recall value, in this case, is still high (= 0.944). This can be linked to the rich information that is provided in the synthetic trace but can also be indicative of the good prediction probabilities of our method: we manage, with the help of the constraints, to infer the smallest set of edges possible (by setting $\lambda = 1$), that is precise and at the same time retrieves almost the entire ground truth graph.

Difference between priors. Choosing $\lambda = 1$ we can tell the difference between the ER and SBM priors - the latter is more efficient in the task of inferring more true positive and less false positive connections between the users, achieving a Precision close to 0.9. This suggests that, in CEM-sbm, the use of the priors p and q in Eq. 25 and Eq. 26 depending on whether an (i, j) user pair belongs in the same community or not, instead of the use of a global parameter r (as in Eq. 16) that is unaware of any community structure, can greatly improve the prediction performance of the optimization when there are communities in the real graph. Additionally, as shown in Table 6 and as we will show later in more detail, CEM-sbm can detect the underlying communities much better than CEM-er: the estimated p, q values of the graph derived by CEM-sbm for $\lambda = 1$ are much closer to reality ($p = 0.063$ and $q = 0.006$), with small relative errors ($\epsilon_p = 0.05$ and $\epsilon_q = 0.143$), while the F1-score is almost optimal (= 0.961). This is a substantial improvement over the F1-score provided by CEM-er (= 0.419).

6.2. Comparison between methods

6.2.1. Propagation subgraph inferred by each model

For a first understanding of the inner workings of each method that we compare with, we can zoom into the propagation graph inferred for a random episode $\mathcal{E}_s = \{22, 17, 18, 81\}$ from the synthetic trace (Fig. 7). Each method receives as an input the first 50,000 lines of the original trace that after preprocessing contains 859 tweets and 12,236 retweets. The ground truth tells us that users 18 and 81 have reposted user 17, who had previously reposted directly the author user 22. As we see in Fig. 7, our method CEM-sbm ($\lambda = 1$) and Peixoto (2019) have inferred the propagation graph of the episode correctly. CEM-er ($\lambda = 1$) has inferred one more false positive edge from 22 to 18 whereas Star and Chain have inferred two false positive edges. Netinf (2012) has inferred only one false positive edge from 18 to 81 whereas the methods by Newman (2018) and Saito et al. (2008) have inferred no edge at all. Of course, this is only one example of a subgraph inferred by each method. We are going to see next the performance and statistics of the entire friendships graphs inferred.

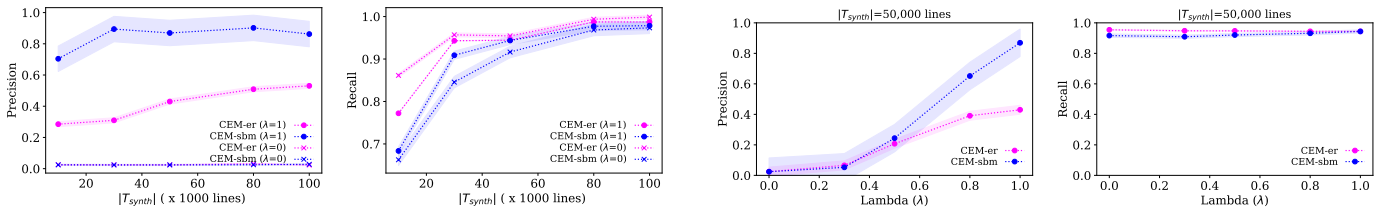
6.2.2. Performance comparison

Precision, Recall, AUC, and graph statistics. Firstly, we are comparing CEM-* with the other methods by looking into the graphs and the performance of each model as described in Section 5.2.2. More specifically, we will compare the performance of each method in terms of Precision, Recall, and AUC. The results are shown in Table 4 and are combined with observations from each graph's statistics, found in Table 5[4].

From there we observe that the two heuristics, **Star and Chain**, give 100% feasible solutions. However, both methods infer graphs with thousands of edges (1,072 and 4,545 edges respectively) and high average out-degrees (10.72 and 46.86) which is very far from reality: the ground truth features only 164 connections with an average out-degree of 1.64. This may result in high Recall and AUC scores but comes at the cost of a very low Precision rate (0.141 and 0.033 respectively, as seen in Table 4). Additionally, both methods infer graphs with very small average shortest paths (< 1.5). In contrast, the ground truth has an average shortest path of 2.57 which is closer to the value that we would expect from a real-world Twitter graph to have. Moreover, Chain infers graphs that are too dense, as seen from its maximum strongly connected component (last column, Table 5: it includes 87% of the users, whereas

[3] https://pypi.org/project/PuLP/

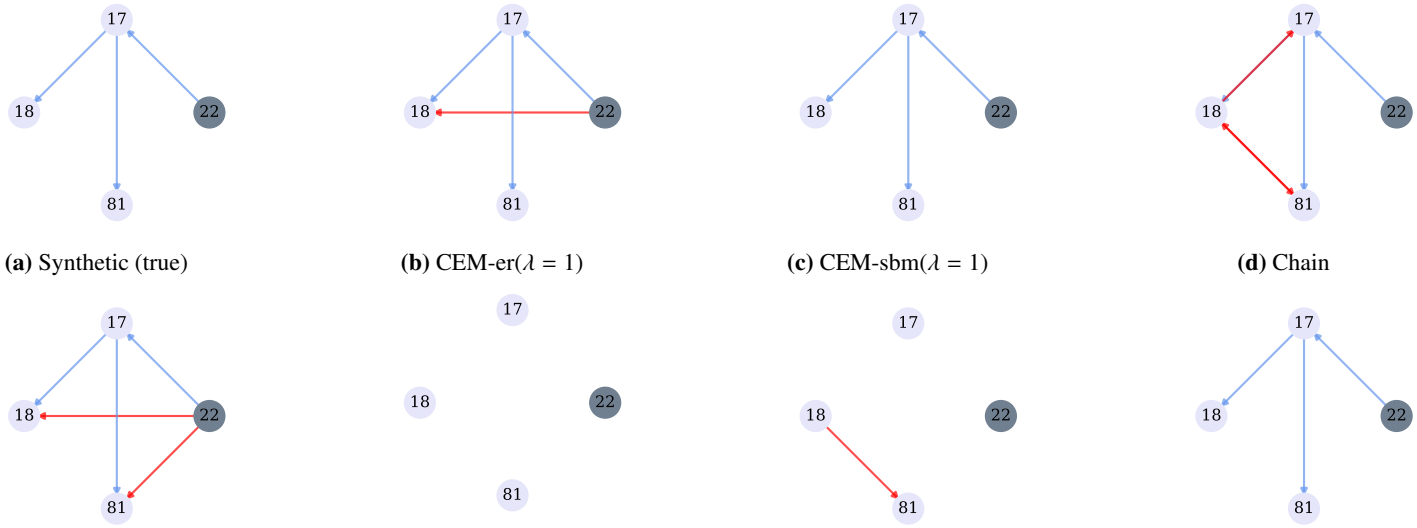
[4] The highest value is marked with boldface and the second highest value is underlined. max scc: maximum strongly connected component.



(a) Results for different sizes of the synthetic trace T_{synth}.

(b) Results for different values of $\lambda \in (0, 1)$ given $|T_{synth}| = 50{,}000$ lines.

Fig. 6: Precision given Recall of CEM-er and CEM-sbm applied on the synthetic dataset.



(a) Synthetic (true)

(b) CEM-er($\lambda = 1$)

(c) CEM-sbm($\lambda = 1$)

(d) Chain

(e) Star

(f) Newman (2018) / Saito et al. (2008)

(g) Netinf (2012)

(h) Peixoto (2019)

Fig. 7: Comparison of the propagation graph inferred by each method for an episode $\mathcal{E}_s = \{22, 17, 18, 81\}$ from the synthetic trace when $|T_{synth}| = 50{,}000$ lines. Each graph shows the real propagation of the tweet s from its author (user 22) to every other user that retweeted it. Blue arrows stand for true positive edges and red arrows stand for false positive ones.

Table 4: Performance of different methods on a synthetic dataset with $|T_{synth}| = 50{,}000$ lines as input.

Performance	Precision	Recall	AUC	runtime (secs)
Star	0.141	**0.956**	0.931	**1.0**
Chain	0.033	0.955	0.752	**1.0**
Saito et al. (2008)	**1.0**	0.051	0.525	3.0
Netinf (2012)	0.159	0.165	0.575	2,199.0
Newman (2018)	0.522	0.450	0.724	2.0
Peixoto (2019)	0.643	0.924	0.958	3,481.0
CEM-er ($\lambda = 0$)	0.024	0.954	0.668	8.0
CEM-er ($\lambda = 1$)	0.430	0.944	0.962	9.0
CEM-sbm ($\lambda = 0$)	0.024	0.916	0.650	1.4
CEM-sbm ($\lambda = 1$)	0.869	0.944	**0.970**	1.5

the actual value is only 11%). The above suggests that, given the synthetic dataset as input, Star and Chain infer graphs that are feasible but demonstrate properties that are far from these of the actual graph, and also, from these of a real-world graph in general.

The method of **Saito et al. (2008)** is 100% precise but produces only 8 edges, a very low number for it to be considered a sufficient solution to our problem. Consequently, it presents a very low feasibility rate: it can only explain 2.33% of the episodes presented in the trace. As a result, its graph properties are far from those of the real graph. For example, the maximum out and in-degrees of the graph are equal to 1, along with the diameter and the average shortest path. Furthermore, the graph inferred by Saito has no strongly connected component and has a very low average out-degree of 0.5.

For the **Netinf (2012)** model, we set in advance $k = 164$ as the number of edges that we want to infer, which is equal to the number of edges of the real graph (however such information will not be available in practice and the authors suggest trying different values of k depending on the desired outcome). As we see, the inferred graph has low feasibility of 34.8% and performs poorly on Precision (= 0.159), Recall (= 0.165), and AUC (=0.575). This is accompanied by weak graph statistics: it has a relatively low maximum out-degree (= 9 whereas the real value is 39), the largest diameter out of all the methods (= 12), and

Table 5: Network statistics of the graph inferred by each method compared to the ground truth for $|T_{synth}| = 50{,}000$.

Inferred network metrics	feasibility(%)	#edges	avg out-degree	max out-degree	max in-degree	diameter	avg shortest path	max scc (%users)
Synthetic graph	100.00	164	1.64	39	15	5	2.57	11
Star	**100.00**	1,072	10.72	78	24	3	1.35	10
Chain	**100.00**	4,545	46.86	75	71	3	1.48	87
Saito et al. (2008)	2.33	8	0.50	1	1	1	1	0
Netinf (2012)	34.80	164*	2.49	9	12	12	4.76	24
Newman (2018)	72.29	138	**1.55**	77	3	1	1	0
Peixoto (2019)	98.02	227	2.34	36	11	10	3.42	19
CEM-er ($\lambda = 0$)	**100.00**	6,175 ± 89	63.66 ± 0.92	94.5 ± 0.17	96	2	1.34	97
CEM-er ($\lambda = 1$)	**100.00**	349 ± 8	3.59 ± 0.09	45.9 ± 2.03	**15.2 ± 0.2**	**5**	2.23	10
CEM-sbm ($\lambda = 0$)	**100.00**	6,141 ± 244	63.31 ± 2.52	91.5 ± 2.95	92.6 ± 3.4	2.1	1.34	97
CEM-sbm ($\lambda = 1$)	**100.00**	**177 ± 11**	1.83 ± 0.12	**41.3 ± 1.65**	11.3 ± 0.15	5.2 ± 0.2	**2.61**	**11.9 ± 0.6**

*chosen a priori

Table 6: Performance of community detection for the synthetic graph with $|T_{synth}| = 50{,}000$ lines.

Label prediction	F1-score
Star	0.350
Chain	0.421
Saito et al. (2008)	–
Netinf (2012)	0.251
Newman (2018)	0.526
Peixoto (2019)	0.731
CEM-er ($\lambda = 1$)	0.419
CEM-sbm ($\lambda = 1$)	**0.961**

Community parameters	$p_{G_{synth}}$	$\lvert\epsilon_p\rvert$	$q_{G_{synth}}$	$\lvert\epsilon_q\rvert$
Synthetic graph	0.060	–	0.007	–
Star	0.148	1.467	0.091	12
Chain	0.682	10.366	0.351	49.142
Saito et al. (2008)	0.500	7.333	–	–
Netinf (2012)	0.285	3.750	0.002	0.714
Newman (2018)	0.043	0.283	**0.007**	**0**
Peixoto (2019)	0.112	0.866	0.006	0.143
CEM-er($\lambda = 1$)	0.085	0.416	0.021	2.000
CEM-sbm($\lambda = 1$)	**0.063**	**0.050**	0.006	0.143

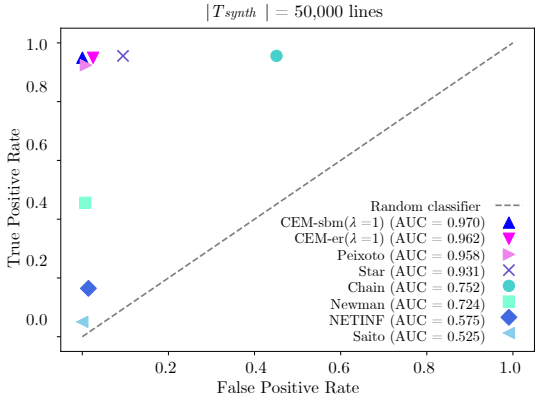


Fig. 8: True Positive related to the False Positive rates of each inference model when applied to the synthetic dataset.

its maximum strongly connected component is more than two times bigger than the real one (it covers 24% of the users).

The method by **Newman (2018)** returns a Precision= 0.522 and Recall= 0.450 which are values close to the output of a random classifier. However, it infers a graph with 138 edges and an average degree of 1.55 which is close to the real numbers. Still, the diameter, maximum in-degree, and average shortest path values are really small compared to the ground truth. Additionally, it presents no strongly connected component. All in all, the graph is neither feasible (feasibility = 72.29%), nor competitive in terms of any performance or statistical metric, which could be due to the fact that it does not consider the hidden paths that exist between users and thus, loses a lot of information that is (indirectly) available in the trace.

The method by **Peixoto (2019)** is the most competitive out of all the above methods, with 98% feasibility, Precision = 0.643 and Recall= 0.924. Additionally, the graph presents some properties that are similar to the ground truth. For example, as we see in Table 5, the derived graph has a maximum out-degree (= 36) whose value is the second closest to the real one (= 39). However, it generates almost 40% more edges and therefore the diameter and the maximum strongly connected component of the graph is almost two times larger than the true one.

To compare with the above, both our methods, CEM-er and CEM-sbm achieve 100% feasibility across all λ values. In addition, CEM-sbm ($\lambda = 1$) achieves the highest performance out of all the methods in terms of Precision, Recall, and AUC (=0.869, 0.944, 0.970 respectively). Furthermore, we see that the graph inferred by CEM-sbm for $\lambda = 1$ has network properties almost identical to the ground truth, followed by the one inferred by CEM-er ($\lambda = 1$).

Optimization runtime. On top of the good prediction and graph statistics results, our algorithm is scalable and achieves running times that are close to the times of the heuristics and far lower than other alternatives (last column, Table 4). CEM-sbm for example runs in less than 1.5 seconds, which is close to the runtimes of Star and Chain. The methods by Newman (2018) and Saito et al. (2008) may have similar runtime, but they lose in accuracy. In contrast, Netinf (2018) and Peixoto (2019) need more than half an hour to converge and still, as we saw above, their results are not as competitive. This makes our optimization method powerful not only in terms of the accuracy of the prediction but also in terms of the time that is needed to reach a result.

ROC curve points of each method. The Precision and Recall point shown in Table 4 for all methods are also visually illustrated on a 2-dimensional True Positive vs False Positive Rate scale (Fig. 8). The upper left corner points correspond to the ideal classifier with AUC= 1; close to that point we find CEM-sbm ($\lambda = 1$), CEM-er ($\lambda = 1$), and Peixoto (2019). Star is close, while the other methods are further away.

Detection of communities. As shown in Table 6, our method CEM-sbm ($\lambda = 1$) achieves the highest F1-score (= 0.961) out of all the methods, followed by the method by Peixoto (= 0.731). Interestingly, the p, q parameters of CEM-sbm ($\lambda = 1$) are close to these of the ground truth ($p_{G_{synth}} = 0.063$ with relative error $|\epsilon_p| = 0.05$ and $q_{G_{synth}} = 0.006$ with relative error $|\epsilon_q| = 0.143$). Among the other methods, regarding p and q, we see that the method by Newman (2018) presents the lowest relative errors regarding the real values (0.283 and 0 respectively).

7. Experiments on the #Élysée2017fr dataset

Next, we will work with real-world data that, as seen in Section 5.1.2, have different properties from the synthetic dataset, making the inference process more challenging.

Table 7: Converged values for parameters α, β given $|T_{elysee}|$ = 5,000,000 lines as input.

CEM-* parameters	$(1-\alpha^*)$	β^*
CEM-er ($\lambda = 0$)	0	1.19e-10
CEM-er ($\lambda = 1$)	1.32e-11	2.46e-12
CEM-sbm ($\lambda = 0$)	5.55e-16	1.07e-10
CEM-sbm ($\lambda = 1$)	0.004	0.001

7.1. Results of our method

Values of parameters. The converged α, β parameters of CEM-er and CEM-sbm can be seen in Table 7. Since all of the α values are close to 1, there is an almost 100% probability that a post spread through an edge that we predicted to exist in the inferred networks. For CEM-er, the smaller value of β, which is almost equal to zero, suggests that there are zero false positive utilized edges. However, in the case of CEM-sbm ($\lambda = 1$), the slightly higher value of $\beta^* = 0.001$ suggests that there is a low, but existing probability, that a tweet passes via an edge that does not appear in the inferred ground truth. As we will see later, this may mean that we have missed some edges and therefore the overall feasibility rate may be (slightly) affected. Likewise, in the same case, the fact that $1 - \alpha^* = 0.004$ means that there is a small probability of false negative utilized edges existing.

Different sizes of input. Figure 9a shows the relation of Precision and Recall given trace sizes that range from 1 to 5 million lines. Again, as was the case with the synthetic data, the more information we have available, the higher the value of Recall will be. These values, however, will still stay at relatively low levels, under 0.1. As seen in Table 2, this is largely due to the fact that only 45.23% of the positive (i, j) edges in the ground truth appear in the trace (i.e., they have $M_{ij} > 0$). The rest of them do not appear in the measurements, therefore it is not possible to infer them given the specific trace we have at hand. Still, we manage to predict thousands of edges that are mostly true positive (as seen from the Precision value). More specifically about Precision, we notice a slight drop as the size of the trace increases. This makes sense, since we infer more edges the more data we get, and therefore we are more likely to make errors. The drop is milder when $\lambda = 1$ and more noticeable when $\lambda = 0$.

Different values of the hyperparameter λ. From Fig. 9b we notice that high values of λ given a constant trace size (= 5 million lines) correspond to higher values of Precision. Here, we observe a trade-off between Precision and Recall, which was not evident in the synthetic dataset: in CEM-sbm for example, the lowest Precision(= 0.213) corresponds to the highest Recall value(= 0.185) when $\lambda = 0$ and a lower Recall value (= 0.074) corresponds to a higher Precision (= 0.478) when λ is set to 1. Therefore, we see that depending on our goal, we can choose to prioritize Precision over Recall and vice-versa. This can be controlled by the correct selection of the hyperparameter λ.

Difference between priors. In contrast to the synthetic dataset case, from the above figures we notice that CEM-er and CEM-sbm present more similar behavior. This is largely due to the properties of the trace itself: we have relatively sparse information on the edges between users that belong to different communities (we observe only 18.95% of the existing inter-edges as seen in Table 2, in contrast to the 98.25% of the positive inter-edges in the case of the synthetic dataset). This makes sense since, in reality, users between different communities interact less often, so it is less likely that they will appear in a trace when we collect it. Therefore, the benefit of using the SBM instead of the ER prior cannot be easily made obvious given the specific trace that we have at hand. Still, the use of the SBM prior provides the highest Recall value (= 0.185, for $\lambda = 0$) and AUC value, (= 0.589, for $\lambda = 0$), which, as we will show later are also the largest values among all compared methods.

7.2. Comparison between methods

We compare the graphs inferred by our two models with the same methods presented before, this time when real-world data is given as input. Given our computational resources, we were not able to run the method by Peixoto (2019) and Netinf (2012) within reasonable timeframes (in < 48 hours), therefore they are left out of the comparison. Table 8 shows the Precision, Recall, and AUC performance of each method, and Table 9 shows the properties of each corresponding graph[5].

7.2.1. Performance comparison

From Tables 8 and 9 we observe that Star and Chain give 100% feasible solutions with Precision equal to 0.446 and 0.262 respectively and Recall values equal to 0.133 and 0.130. However, their graph statistics resemble less these of the real graph: **Star** infers 463,290 edges, with max out-degree equal to 2,524 and max in-degree equal to 1,069. We consider these values quite high, given the number of edges inferred (they are comparable to the ground truth which has three times the number of edges of Star) and that's why we consider it less trustworthy. This result is expected due to the heuristic method of inferring the edges, which connects directly the author of a post to its reposters.

The graph by **Chain**, as seen in the last column of Table 9, has the highest maximum strongly connected component (it includes 95.34% of all users), which is bigger than the corresponding size in the real graph (= 93.28%). Given that the inferred graph by Chain is half the size of the real graph, this high percentage suggests that it is more densely connected than we would expect from a real graph. What is more, in a real-world graph, most nodes have a relatively small degree, but some of them will have a noticeably larger degree, being connected to many other nodes. However, in Chain, we do not notice this phenomenon.

As was the case in the synthetic dataset evaluation, the method of Saito et al. (2008) generates only a few edges (= 768) and is therefore not feasible. The method of **Saito et al. (2008)** may be again relatively precise, but presents no strongly connected component, has a very low average out-degree (= 0.5) and an abnormally high diameter (= 8), given the size of the graph. The above shows that the graph inferred by this method is very sparse and does not resemble the real-world graph in question.

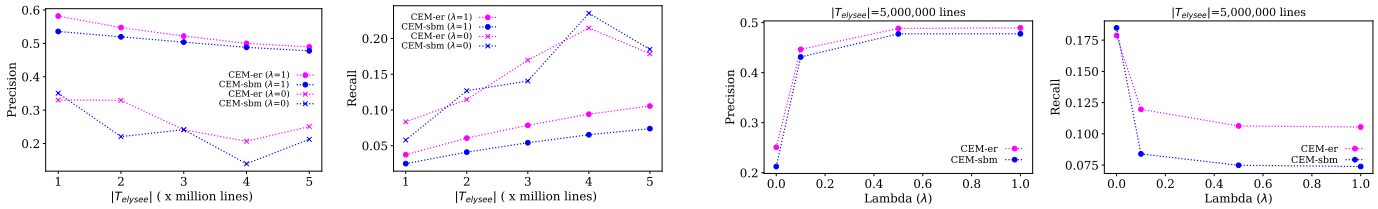
Likewise, the model by **Newman (2018)** is not feasible, but in this case, seems more competitive in terms of the Precision metric (= 0.464). However, its large diameter (= 12.7) given the size of the inferred graph (5 times smaller than the real graph which has a diameter = 11) prevents us from selecting it as a realistic option.

Compared with the above methods, our algorithm CEM-*, presents the highest values in terms of every metric: Precision, Recall, or AUC. This can be regulated either by choosing a value close to $\lambda = 0$, that returns the highest number of nodes (>1,100,000) and therefore a high Recall (=0.178), for CEM-sbm ($\lambda = 0$)) but lower Precision, or by choosing a value closer to $\lambda = 1$ that returns less than 340,000 nodes (for both priors) and therefore a lower Recall but a high Precision (=0.489, CEM-er ($\lambda = 1$)). When it comes to the statistics of the graph, its diameter stays close to the real value (= 11). The same is true for the average shortest path. This illustrates that the two best values from each category are in favor of our CEM-* method.

Optimization runtime. We verify from the runtime column of Table 8 that our model is scalable since we manage to solve an optimization problem with 6,922,990 unknowns and 1,605,059 constraints in only a couple of hours. We achieve this not only by formulating the inference as a linear optimization problem but also by taking advantage of powerful optimization solvers that are publicly available (in our case, the Gurobi solver). On the other hand, the methods by Saito et al. and Newman present fast computation times (342 and 25 seconds) but, as we have shown, they present less competitive results in terms of feasibility or performance.

ROC curve points of each method. Again, on the upper left corner of the True Positive vs False Positive Rate figure (Figure 10), we find our methods CEM-sbm ($\lambda = 0$) and CEM-er ($\lambda = 0$). Star and Chain

[5] N/A in the Tables refers to results not being available after 48 hours.



(a) For different sizes of the trace T_{elysee} and $\lambda \in \{0, 1\}$.

(b) For different values of $\lambda \in (0, 1)$ given $|T_{elysee}| = 5$ mil. lines.

Fig. 9: Precision given Recall of CEM-er and CEM-sbm applied on #Élysée2017fr.

Table 8: Performance of each method for the #Élysée2017fr dataset.

Performance	Precision	Recall	AUC	runtime(secs)
Star	0.446	0.133	0.565	**1**
Chain	0.262	0.130	0.563	**1**
Saito et al. (2008)	0.199	0.0001	0.500	342.00
Netinf (2012)	N/A	N/A	N/A	N/A
Newman (2018)	0.464 ± 0.031	0.066 ± 0.001	0.533 ± 0.001	25.00
Peixoto (2019)	N/A	N/A	N/A	N/A
CEM-er ($\lambda = 0$)	0.251	**0.179**	0.586	37,721.00
CEM-er ($\lambda = 1$)	0.489	0.105	0.552	35,552.00
CEM-sbm ($\lambda = 0$)	0.213	0.185	**0.589**	44,016.00
CEM-sbm ($\lambda = 1$)	**0.478**	0.074	0.537	88,504.00

Table 9: Network statistics of the graphs inferred by each method compared to the ground truth graph for $|T_{elysee}|$ = 5,000,000 lines.

Inferred network metrics	feasibility(%)	#edges	avg out-degree	max out-degree	max in-degree	diameter	avg shortest path	max scc (% users)
Ground-truth	49.00	1,555,718	136.42	5,004	1,853	11	2.82	93.28 (10,747)
Star	**100.00**	463,290	40.25	**2,524**	1,069	**12**	3.70	66.05 (7,610)
Chain	**100.00**	768,122	66.73	1,122	1,256	8	3.04	**95.34 (10,984)**
Saito et al. (2008)	0.55	786	0.54	2	10	8	1.11	0
Netinf (2012)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Newman (2018)	37.24	237,063	22.43	1,206 ± 127	558	12.7	4.32	52.57 (6,057)
Peixoto (2019)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
CEM-er ($\lambda = 0$)	**100.00**	1,108,079	96.26	2,336	1,262	9	3.06	80.31 (9,252)
CEM-er ($\lambda = 1$)	**100.00**	335,289	29.13	2,291	790	12	3.82	66.07 (7,612)
CEM-sbm ($\lambda = 0$)	**100.00**	**1,353,432**	**117.58**	1,364	**1,609**	8	**2.95**	82.50 (9,505)
CEM-sbm ($\lambda = 1$)	99.37	240,893	20.97	955	775	**11**	3.58	72.81 (8,388)

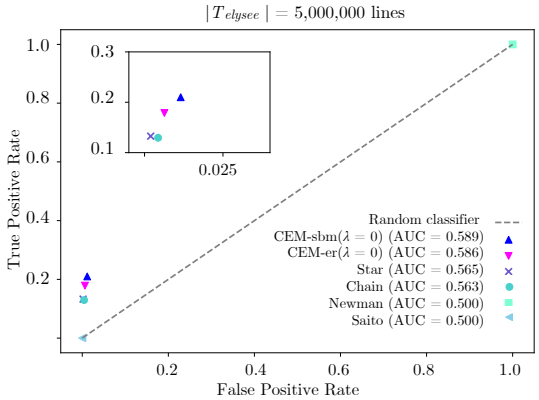


Fig. 10: True Positive related to the False Positive rates of each inference model when applied on #Élysée2017fr.

are a bit lower, and the other methods are further away. This suggests that our model has the highest capacity to differentiate between the two classes (existing and non-existing edges) among all the other methods (and is also why we have the highest AUC values, as seen in Table 8).

Detection of communities. As shown in Table 10, our methods CEM-er ($\lambda = 0$) and CEM-er ($\lambda = 1$) achieve a high F1-score (= 0.888 and 0.887), similarly to Newman's method (= 0.888) and Chain (= 0.889). Chain's high performance does not surprise us in this case since Chain favors the creation of communities all while inferring a very high number of edges compared to other methods. Despite this, all the p parameters estimated on the graphs by each method are far from the real ground truth value. This was expected since we are missing substantial information on how edges interact between different communities and we may therefore be overestimating the value of p while underestimating q. Still, our method for $\lambda = 1$ has the lowest relative error on the p parameter (7.58 for CEM-er and 5.17 for CEM-sbm) along with Newman that has an $\epsilon_p = 6.75$.

7.3. Controlling feasibility through β

As expected, since 2017 (the year that the dataset was created), some Twitter profiles have been deleted or set to private. In addition, users may have retweeted a tweet/episode outside the scope of their followees (e.g., through Twitter search, recommendation algorithms, Twitter trends, etc.). As a result, the #Élysée2017fr trace is not 100% feasible given the ground truth friendship graph. In other words, the current view of the friendship graph does not explain all the episodes in the selected trace; in fact, it can only explain 49% of them.

Table 10: Performance of community detection for the real-world graph with $|T_{elysee}|$ = 5,000,000 lines.

Label prediction	F1-score
Star	0.858
Chain	**0.889**
Saito et al. (2008)	0.447
Netinf (2012)	N/A
Newman (2018)	0.888
Peixoto (2019)	N/A
CEM-er ($\lambda = 0$)	0.888
CEM-sbm ($\lambda = 0$)	0.880
CEM-er ($\lambda = 1$)	0.887
CEM-sbm ($\lambda = 1$)	0.878

Community parameters	p_G	$\|\epsilon_p\|$	q_G	$\|\epsilon_q\|$
Ground-truth	0.0012	N/A	0.0445	N/A
Star	0.0143	10.92	0.0003	0.99
Chain	0.0236	18.67	0.0004	0.99
Saito et al. (2008)	0.3834	318.5	N/A	N/A
Netinf (2012	N/A	N/A	N/A	N/A
Newman (2018	0.0093	6.75	5e-05	1
Peixoto (2019	N/A	N/A	N/A	N/A
CEM-er ($\lambda = 0$)	0.0346	27.83	0.0005	0.99
CEM-sbm ($\lambda = 0$)	0.0425	34.42	**0.0006**	**0.99**
CEM-er ($\lambda = 1$)	0.0103	7.58	0.0002	1
CEM-sbm ($\lambda = 1$)	**0.0074**	**5.17**	0.0001	1

Table 11: Performance of CEM-* given constant values of parameter β for #Élysée2017fr.

Performance given β	Precision	Recall	AUC	feasibility(%)
CEM-er ($\lambda = 1$)	0.489	**0.105**	**0.552**	100.0
CEM-er ($\lambda = 1$) ($\beta = 0.5$)	0.592	0.060	0.530	66.96
CEM-er ($\lambda = 1$) ($\beta = 0.6$)	0.604	0.052	0.526	61.21
CEM-er ($\lambda = 1$) ($\beta = 0.7$)	**0.619**	0.040	0.520	**52.78**
CEM-sbm ($\lambda = 1$)	0.478	0.074	0.537	99.37
CEM-sbm ($\lambda = 1$) ($\beta = 0.5$)	0.552	0.054	0.527	71.86
CEM-sbm ($\lambda = 1$) ($\beta = 0.6$)	0.558	0.048	0.524	65.65
CEM-sbm ($\lambda = 1$) ($\beta = 0.7$)	0.566	0.041	0.520	58.00

We can therefore control the feasibility of our result to match the feasibility of the trace given the ground truth through the parameter β: for an inferred graph to be feasible, we want the false positive utilization rate β, i.e. the average number of inferred edges that pass through an edge that does not exist in the inferred graph, to be as close to 0 as possible. If β is close to a non-zero value, it means that there is a $\beta > 0$ probability that influence has happened through a nonexistent edge in the inferred graph and therefore some episodes may be left unexplained. Consequently, we can set β equal to a constant - instead of updating it through Eq. 15 or 24 - whose value depends on the feasibility that we wish the outcome to have. Hence, we will examine the relation of the inferred graph to the ground truth given different constant values of β. In general, we expect the inferred graph to be more precise when the feasibility rate is close to this of the ground truth graph (= 49%).

First of all, as we show in Table 11 when β increases feasibility decreases. For example, when $\beta = 0.7$ the feasibility of the trace given the inferred graph is 52.78% and 58% for CEM-er and CEM-sbm respectively. We note that the value of β changes only the overall number of edges inferred, which indirectly affects the number of episodes that are explained in the trace. Furthermore, as β increases, and hence feasibility decreases, we get closer to the actual 50% feasibility and precision improves. We should underline that when the feasibility rate falls lower than 50% (for $\beta > 0.7$), Precision falls dramatically since the algorithm starts inferring edges randomly, without really respecting the constraints.

8. Evaluation with no ground truth

Overall, we notice that feasibility is proved beneficial and can increase the quality of the inferred graph. For example, the methods with the lowest feasibility rates (Saito et al., Netinf, Newman) infer graphs with low predictive quality and present statistics that are far from those of the real-world graph. The benefit of CEM-* over other methods is especially apparent when we have collected sufficient data between edges, as was the case in the synthetic dataset case. Consequently, when the underlying friendship graph is not available, which is often the case in graph reconstruction problems, the feasibility rate of the inferred graph could be an effective indicator of a method's performance.

However, feasibility is not a sufficient condition for better prediction results. As we see in the case of Star, Chain, and CEM-* for $\lambda = 0$, a 100% feasibility rate cannot guarantee a precise result. Moreover, as we showed, we can use empirical values about how much feasibility to require in the inferred graph based, for example, on how old the trace is, or how often users retweet outside of their connections, e.g., using recommendations. On top of feasibility, we could look into the inferred graph's statistics and evaluate to what extent they are similar to these of a general, real-world graph. Usual indicators of such real-world properties are the average degree, the diameter, the average shortest path, and the strongly connected components of the graph.

9. Conclusions

As we observed above, CEM-* successfully produces feasible graphs that are closer to reality when compared to heuristic and state-of-the-art methods. We validated the results both on synthetic and real-world traces, using two different graph priors, Erdős-Rényi (ER) and Stochastic Block Model (SBM), and noticed that CEM-* produces results that in most cases return the two most accurate values among all chosen compared metrics, and does so significantly faster than the state-of-the-art. Moreover, by selecting values between 0 and 1 for the hyperparameter λ, we can control the trade-off between the Precision and Recall of the result.

When comparing the effect of the two graph priors, we notice that the use of SBM can improve inference accuracy and community detection. The contribution of SBM is more apparent when we have sufficient information on how nodes interact between different communities.

Furthermore, we observe that feasible graphs are usually closer to the underlying graph compared to non-feasible graphs. When the trace is 100% feasible given the ground truth, as is the case in the synthetic dataset, we find that feasibility is a necessary condition for the inferred graph to be as close to reality as possible. In the case of the #Élysée2017fr graph, where the trace is only 50% feasible, we observed that Precision improves as we force the feasibility of the inferred graph to be closer to the real percentage (through the parameter β). We conclude therefore that, for higher Precision, the feasibility of the trace given the inferred graph should match the feasibility of the trace given

the true graph. However, if we cannot be sure about the ground truth's feasibility, we still suggest starting working with $\beta = 0$, since, as we saw, it still returns better, and more realistic networks than other inference methods. Keep in mind, that as we saw in the case of Star and Chain, feasibility is not a sufficient condition for the accuracy of the result: the graphs inferred by both these methods are 100% feasible but present some extreme properties (e.g., large diameter, low maximum degree) that make the results less trustworthy.

We should note here that our method works with a specific trace structure that is based on the data that most social media platforms currently offer. In future work, we plan to apply our method and constraints to other types of data and graph inference cases to adapt to a wider range of domains (such as biology, epidemics, etc).

Acknowledgements. An earlier version of this paper was presented at the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 09-11 November 2021 (virtual) ASONAM 2021. This work is funded by the ANR (French National Agency of Research) by the "FairEngine" project under grant ANR-19-CE25-0011.

References

Blondel V, Guillaum JL, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment

Bourigault S, Lamprier S, Gallinari P (2016) Representation learning for information diffusion through social networks: an embedded cascade model. In: Proceedings of the 9th ACM International Conference on Web Search and Data Mining pp. 573-582

Daley DJ, Gani J (1999) Epidemic modelling: an introduction. Cambridge University Press

Daneshmand H, Gomez-Rodriguez M, Song L, Schoelkopf B (2014) Estimating Diffusion Network Structures: Recovery Conditions, Sample Complexity & Softthresholding Algorithm. In: Proceedings of the 31st International Conference on Machine Learning (ICML)

Dempster AP, Laird NM, Rubin DB (1977) Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society.

Firestone SM, Hayama Y, Lau MSY, Yamamoto T, Nishi T, Bradhurst RA, Demirhan H, Stevenson MA, Tsutsui T (2020) Transmission network reconstruction for foot-and-mouth disease outbreaks incorporating farm-level covariates. PLoS One

Fraisier O, Cabanac O, Pitarch Y, Besançon R, Boughanem M (2018) #Elysee2017fr: The 2017 French Presidential Campaign on Twitter. In: Proceedings of the 12th International AAAI Conference on Web and Social Media

Friedman N, Linial M, Nachman I, Pe'er D, Using bayesian networks to analyze expression data. In: Journal of Computational Biology Vol. 7 pp. 601–620

Giesecke K, Schwenkler G, Sirignano J A (2020) Inference for large financial systems. Mathematical Finance pp. 3-46

Giovanidis A, Baynat B, Magnien C, Vendeville A (2021) Ranking online social users by their influence. IEEE/ACM Transactions on Networking

Gomez-Rodriguez M, Leskovec J, Krause A (2012) Inferring networks of diffusion and influence. ACM Transactions on Knowledge Discovery from Data (TKDD)

Goyal A, Bonchi F, Lakshmanan LV (2010) Learning influence probabilities in social networks. InProceedings of the third ACM international conference on Web search and data mining pp. 241-250

Harris JW, Stöcker H (1998) Handbook of mathematics and computational science. Springer Science & Business Media

He X, Liu Y (2017) Not enough data? Joint inferring multiple diffusion networks via network generation priors. InProceedings of the Tenth ACM International Conference on Web Search and Data Mining pp. 465-474

Holland PW, Laskey KB, Leinhardt S (1983) Stochastic blockmodels: First steps. Social networks pp. 109-137

Jin W, Qu M, Jin X, Ren X (2019) Recurrent event network: Autoregressive structure inference over temporal knowledge graphs

Lagnier C, Denoyer L, Gaussier E, Gallinari P (2013) Predicting information diffusion in social networks using content and user's profiles. In: European conference on information retrieval pp. 74-85

Le CM, Levin K, Levina E (2018) Estimating a network from multiple noisy realizations. Electronic Journal of Statistics pp. 4697-4740

Lokhov A (2016) Reconstructing parameters of spreading models from partial observations. Advances in Neural Information Processing Systems pp. 3467–3475

Newman ME (2018) Network structure from rich but noisy data. Nature Physics Vol. 14 pp. 67-75

Newman ME (2018) Estimating network structure from unreliable measurements. Physical Review E

Papanastasiou E, Giovanidis A (2021) Bayesian inference of a social graph with trace feasibility guarantees. In: Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '21)

Peel L, Peixoto TP, De Domenico M (2022) Statistical inference links data and theory in network science. Nature Communications pp. 1-15

Peixoto TP (2019) Network reconstruction and community detection from dynamics. Physical Review Letters

Saito K, Nakano R, Kimura M (2008) Prediction of information diffusion probabilities for independent cascade model. In: International conference on knowledge-based and intelligent information and engineering systems pp. 67-75

Wang Z, Chen C, Li W (2019) Information diffusion prediction with network regularized role-based user representation learning. ACM Transactions on Knowledge Discovery from Data (TKDD) pp. 1-23

Wu X, Kumar A, Sheldon D, Zilberstein S (2013) Parameter learning for latent network diffusion. In Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence pp. 2923–2930

Wu J, Xia J, Gou F (2022) Information transmission mode and IoT community reconstruction based on user influence in opportunistic social networks. Peer-to-Peer Networking and Applications pp. 1398-1416

Zhang Y, Lyu T, Zhang Y (2018) Cosine: Community-preserving social network embedding from information diffusion cascades. In: Proceedings of the AAAI conference on artificial intelligence Vol. 32

Zhang X, Zhang Z K, Wang W, Hou D, Xu, J, Ye X, Li S (2021) Multiplex network reconstruction for the coupled spatial diffusion of infodemic and pandemic of COVID-19. International Journal of Digital Earth Vol. 4 pp. 401-423

Appendix A: CEM-er

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 (10):

$$\mathbb{E}[\log P(\theta \mid \mathcal{T})] \geq \mathbb{E}[\sum_{\mathbf{A}} q(\mathbf{A}) \log \frac{P(\mathbf{A}, \theta \mid \mathcal{T})}{q(\mathbf{A}}]$$
$$= \sum_{\mathbf{A}} q(\mathbf{A})(\mathbb{E}[\log P(\mathbf{A}, \theta \mid \mathcal{T})] - \log q(\mathbf{A})). \quad \text{(A.1)}$$

To find $\mathbb{E}[\log P(\mathbf{A}, \theta \mid \mathcal{T})]$, we replace (9) into (8). Setting $\Gamma = P(\theta)/P(\mathcal{T})$, the expectation of the log of (8) becomes:

$$\mathbb{E}[\log P(\mathbf{A}, \theta \mid \mathcal{T})] = log\Gamma + \sum_{i \neq j} \Big[A_{ij}\Big(\log \rho + \mathbb{E}[Y_{ij}] \log \alpha +$$
$$+(M_{ij} - \mathbb{E}[Y_{ij}]) \log (1-\alpha)\Big) + (1 - A_{ij})\Big(\log(1-\rho) +$$
$$+\mathbb{E}[Y_{ij}] \log \beta + (M_{ij} - \mathbb{E}[Y_{ij}]) \log (1-\beta)\Big)\Big]. \quad \text{(A.2)}$$

Then, by replacing (7) into (A.2), and then (A.2) into (A.1), we get:

$$\mathbb{E}[\log P(\theta \mid \mathcal{T})] \geq \sum_{\mathbf{A}} q(\mathbf{A}) \log \frac{D_{ij}}{q(\mathbf{A})} \quad \text{(A.3)}$$

$$\text{where, } D_{ij} = \Gamma \prod_{i \neq j} \Big[\rho \alpha^{M_{ij}\sigma_{ij}} (1-\alpha)^{M_{ij}(1-\sigma_{ij})}\Big]^{A_{ij}}$$
$$\times \Big[(1-\rho)\beta^{M_{ij}\sigma_{ij}} (1-\beta)^{M_{ij}(1-\sigma_{ij})}\Big]^{1-A_{ij}}. \quad \text{(A.4)}$$

For the M-step of the EM algorithm, the function that we want to maximize is $\mathbb{E}[\log P(\theta \mid \mathcal{T})]$. 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 (11), 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).

Solution 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 (11) is:

$$q(\mathbf{A}) = \frac{D_{ij}}{\sum_{\mathbf{A}} D_{ij}}, \quad \text{(A.5)}$$

which leads us to Eq. (13).

Solution with respect to α, β, ρ. To maximize the right-hand side of (11) in terms of parameter α we differentiate it with respect to α and then setting it equal to zero (while holding σ_{ij}, q constant):

$$\sum_{i \neq j} Q_{ij} M_{ij} \left(\frac{\sigma_{ij}}{\alpha} - \frac{1-\sigma_{ij}}{1-\alpha}\right) = 0. \quad \text{(A.6)}$$

After rearranging, we get the updates shown in Eq. (15), and we repeat likewise for β and ρ.

Solution with respect to σ_{ij}. 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 (11), the problem becomes equivalent to maximizing:

$$\sum_{\mathbf{A}} q(\mathbf{A}) \sum_{i \neq j} \sigma_{ij} M_{ij} \left(A_{ij} \log \frac{\alpha}{1-\alpha} + (1 - A_{ij}) \log \frac{\beta}{1-\beta}\right)$$
$$= \sum_{i \neq j} \sigma_{ij} M_{ij} \left(Q_{ij} \log \frac{\alpha}{1-\alpha} + (1 - Q_{ij}) \log \frac{\beta}{1-\beta}\right). \quad \text{(A.7)}$$

This leads us to the constrained optimization problem of Eq. (17).

Appendix B: CEM-sbm

For the E-step of the EM algorithm, we modify the Newman algorithm by taking the expectation over the set of random variables Y_{ij} at both sides of (20):

$$\mathbb{E}[\log P(\theta \mid \mathcal{T})] \geq \mathbb{E}[\sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) \log \frac{P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{T})}{q(\mathbf{A}, \mathbf{g})}]$$
$$= \sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g})(\mathbb{E}[\log P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{T})] - \log q(\mathbf{A}, \mathbf{g})). \quad \text{(B.1)}$$

To find $\mathbb{E}[\log P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{T})]$, we replace (19) into (18). Setting $\Gamma = P(\theta)/P(\mathcal{T})$, the expectation of the log of (18) becomes:

$$\mathbb{E}[\log P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{T})] = log\Gamma + \sum_{\substack{i \neq j \\ g_i = g_j}} \Big[A_{ij}\Big(\log p + \mathbb{E}[Y_{ij}] \log \alpha +$$
$$+(M_{ij} - \mathbb{E}[Y_{ij}]) \log (1-\alpha)\Big) + (1 - A_{ij})\Big(\log(1-p) +$$
$$+\mathbb{E}[Y_{ij}] \log \beta + (M_{ij} - \mathbb{E}[Y_{ij}]) \log (1-\beta)\Big)\Big] + \sum_{\substack{i \neq j \\ g_i \neq g_j}} \Big[A_{ij}\Big(\log q +$$
$$+\mathbb{E}[Y_{ij}] \log \alpha + (M_{ij} - \mathbb{E}[Y_{ij}]) \log (1-\alpha)\Big) + (1 - A_{ij})\Big(\log(1-q) +$$
$$+\mathbb{E}[Y_{ij}] \log \beta + (M_{ij} - \mathbb{E}[Y_{ij}]) \log (1-\beta)\Big)\Big]. \quad \text{(B.2)}$$

By replacing (7) into (B.2), and then (B.2) into (B.1), we get:

$$\mathbb{E}[\log P(\theta \mid \mathcal{T})] \geq \sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) \log \frac{D(\mathbf{A}, \mathbf{g})}{q(\mathbf{A}, \mathbf{g})}, \quad \text{(B.3)}$$

where,

$$D(\mathbf{A}, \mathbf{g}) = \Gamma \prod_{\substack{i \neq j \\ g_i = g_j}} \Big[p \alpha^{M_{ij}\sigma_{ij}} (1-\alpha)^{M_{ij}(1-\sigma_{ij})}\Big]^{A_{ij}}$$
$$\Big[(1-p)\beta^{M_{ij}\sigma_{ij}} (1-\beta)^{M_{ij}(1-\sigma_{ij})}\Big]^{1-A_{ij}} \prod_{\substack{i \neq j \\ g_i \neq g_j}} \Big[q \alpha^{M_{ij}\sigma_{ij}} (1-\alpha)^{M_{ij}(1-\sigma_{ij})}\Big]^{A_{ij}}$$
$$\Big[(1-q)\beta^{M_{ij}\sigma_{ij}} (1-\beta)^{M_{ij}(1-\sigma_{ij})}\Big]^{1-A_{ij}}. \quad \text{(B.4)}$$

For the M-step of EM, we maximize the expectation $\mathbb{E}[\log P(\theta \mid \mathcal{T})]$ as we did in the CEM-er prior.

Solution with respect to $q(\mathbf{A}, \mathbf{g})$. We notice that the choice of $q(\mathbf{A}, \mathbf{g})$ that achieves equality (i.e. maximizes the right-hand side) in (B.3) is:

$$q(\mathbf{A}, \mathbf{g}) = \frac{D(\mathbf{A}, \mathbf{g})}{\sum_{\mathbf{A}} D(\mathbf{A}, \mathbf{g})}. \quad \text{(B.5)}$$

From (B.5), in a similar fashion to Newman's method [Eq. (13), 20], and because Γ cancels out, we get:

$$q(\mathbf{A}, \mathbf{g}) = \prod_{i \neq j, (g_i = g_j)} Q_{ij}(g_i, g_j)^{A_{ij}} (1 - Q_{ij}(g_i, g_j))^{1-A_{ij}}$$
$$\prod_{i \neq j, (g_i \neq g_j)} Q_{ij}(g_i, g_j)^{A_{ij}} (1 - Q_{ij}(g_i, g_j))^{1-A_{ij}}. \quad \text{(B.6)}$$

Hence, given Eq. (B.4), the values of Q_{ij} are found to be the ones in Eq. (21) and (22).

Our goal is to find the unknown parameters $\theta = \{\alpha, \beta, p, q, \boldsymbol{\sigma}\}$ that maximize the right-hand size of (B.3), given the maximising distribution for $q(\mathbf{A}, \mathbf{g})$ in (B.5), hence given the values of $Q_{ij}(g_i, g_j)$ in (B.6).

Solution with respect to α, β, p, q. To maximize the right-hand side of (B.3) in terms of parameter α, we differentiate the

equation with respect to α and we set it equal to zero (while holding the rest of the parameters θ constant):

$$\sum_{i\neq j} Q_{ij}(g_i, g_j) M_{ij} \left(\frac{\sigma_{ij}}{\alpha} - \frac{1-\sigma_{ij}}{1-\alpha} \right) = 0. \tag{B.7}$$

After rearranging, we get the value in Eq. (23). By repeating the same procedure for β, we get Eq. (24). Likewise, differentiating the r.h.s. of (B.3) with respect to p and then setting it equal to zero we get:

$$\sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) \sum_{\substack{i\neq j \\ g_i = g_j}} \left(\frac{A_{ij}}{p} - \frac{1-A_{ij}}{1-p} \right) = 0. \tag{B.8}$$

This is how we get the updates for p in Eq. (25), and, likewise, for q in Eq. (26).

Solution with respect to σ_{ij}. If we take into account that $Q_{ij}(g_i, g_j) = \sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) A_{ij}$ and also that $\sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) = 1$, by rearranging the right-hand side of (B.3), the problem becomes equivalent to maximizing:

$$\begin{aligned} &\sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) \sum_{i\neq j} \sigma_{ij} M_{ij} \left(A_{ij} \log \frac{\alpha}{1-\alpha} + (1 - A_{ij}) \log \frac{\beta}{1-\beta} \right) \\ = &\sum_{i\neq j} \sigma_{ij} M_{ij} \left(Q_{ij}(g_i, g_j) \log \frac{\alpha}{1-\alpha} + (1 - Q_{ij}(g_i, g_j)) \log \frac{\beta}{1-\beta} \right). \end{aligned} \tag{B.9}$$

This leads us to the optimization problem of Eq. (27) through which we can find the σ_{ij} values.