# Learning Clusters through Information Diffusion

Liudmila Prokhorenkova
Moscow Institute of Physics and Technology
Yandex
Moscow, Russia
ostroumova-la@yandex.ru

Alexey Tikhonov
Yandex
Berlin, Germany
altsoph@gmail.com

Nelly Litvak
University of Twente
Eindhoven University of Technology
Enschede, The Netherlands
n.litvak@utwente.nl

## ABSTRACT

When information or infectious diseases spread over a network, in many practical cases, one can observe when nodes adopt information or become infected, but the underlying network is hidden. In this paper, we analyze the problem of finding communities of highly interconnected nodes, given only the infection times of nodes. We propose, analyze, and empirically compare several algorithms for this task. The most stable performance, that improves the current state-of-the-art, is obtained by our proposed heuristic approaches, that are agnostic to a particular graph structure and epidemic model.

## KEYWORDS

Community detection; information propagation; information cascades; network inference; likelihood optimization

**ACM Reference Format:**
Liudmila Prokhorenkova, Alexey Tikhonov, and Nelly Litvak. 2019. Learning Clusters through Information Diffusion. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313560

## 1 INTRODUCTION

Diffusion processes in networks include spreading of infectious diseases [14], spread of computer viruses [36], promotion of products via viral marketing [16], propagation of information [31], etc. In many practical scenarios one can observe when nodes become infected but the underlying network is hidden. For instance, during an epidemic, a person becomes ill but cannot tell who infected her; in viral marketing, we observe when customers buy products but not who influenced their decisions; in Twitter, for each retweet only information about the root node of the cascade is available. The problem of hidden network inference received a lot of attention recently [7, 27, 29, 30]. While these papers aim to recover the actual network connections, in many applications only some global properties of the underlying network are important. For example, in viral marketing, one may wish to find the most influential users, while for recommendation systems one may look for groups of users with similar preferences.

In this paper, we analyze the problem, which recently attracted interest in the literature [5, 26], of inferring *community structure* of a given network based solely on *cascades* (e.g., information or epidemic) propagating through this network. Communities are groups of highly interconnected nodes with relatively few edges joining nodes of different groups [9], such as groups by interests in social networks or pages on related topics on the Web. Discovering communities is one of most prominent tasks in network analysis. Compared to the traditional community detection, our work is quite different, because we do not have the network available to us, we have only cascade data observed on this network. For each cascade, we observe only infected nodes and their infection timestamps.

We propose and analyze several algorithms to solve this problem and compare them to the state-of-the-art. The contributions of this paper are the following. First, we give a systematic treatment to the new problem of inferring community structure based on cascade data, as stated formally in Section 3. Second, we propose two types of approaches for this task: based on likelihood maximization under specific model assumptions, and based on clustering of a surrogate network (Section 4). Third, we conduct extensive experiments on a variety of real-world networks (see Section 5). We conclude that the most stable performance is obtained by our proposed heuristics that are agnostic to a particular graph structure and epidemic model. These heuristics work equally well on different networks, for epidemics of different types.

This paper is published under the Creative Commons Attribution 4.0 International (CC-BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.
*WWW '19, May 13–17, 2019, San Francisco, CA, USA*
© 2019 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC-BY 4.0 License.
ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313560

## 2 RELATED WORK

### 2.1 Network Inference from Cascades

A series of recent papers addressed the following task: by observing infection (activation) times of nodes in several cascades, infer the edges of the underlying network. NetInf algorithm developed in [11] is based on the maximization of the likelihood of the observed cascades, based on a specific epidemic model. To make optimization feasible, for each cascade NetInf considers only the most likely propagation tree. This was later improved by MultiTree [30] that includes all directed trees in the optimization and has better performance if the number of cascades is small. NetRate algorithm [28] infers not only edges, but also infection rates. NetRate builds on an epidemic model that is more tractable for theoretical analysis (we describe this model in Section 3.2). For this model the likelihood optimization problem turns out to be convex. ConNIe algorithm [21] also uses convex optimization, with the goal to infer transmission probabilities for all edges. In [13, 29], it is additionally assumed that the underlying network is not static and the proposed algorithm InfoPath provides on-line estimates of the structure and temporal dynamics of the hidden network. KernelCascade [8] also extends NetRate, here the authors avoid assumptions about a particular parametric form of the influence function. The DANI algorithm [27] is interesting because it explicitly accounts for the community structure to enhance the inference of networks' edges. There are some other network inference algorithms not covered here, see, e.g., [7, 12, 22, 32, 34, 35, 37, 38, 40].

## 2.2 Community Inference from Cascades

To the best of our knowledge, the paper [4] extended in [5] for the first time addressed the problem of community detection given cascades. In [4, 5], the cascade model includes influence of individual nodes, and a membership level of a node in each community is inferred using the maximum likelihood approach. The authors propose two algorithms: C-IC takes into account only participation of a node in a cascade; C-Rate includes the time stamps, but limits the node's influence by its own community. Recently, [26] proposed an alternative maximum likelihood approach, which exploits the Markov property of the cascades. As an input, similarity scores of node pairs are computed, based on their joint participation in cascades. The R-CoDi algorithm in [26] starts with a random partition, while D-CoDi starts with a partition obtained by DANI [27]. We use all four mentioned algorithms as our baselines.

## 3 PROBLEM SETUP

### 3.1 General Setup

*Cascades.* We observe a set of cascades $\mathbf{C} = \{C_1, \dots, C_r\}$ that propagate on a latent undirected network $G = (V, E)$ with $|V| = n$ nodes and $|E| = m$ edges. Each cascade $C \in \mathbf{C}$ is a record of observed node activation times, i.e., $C = \{(v_i, t^C_{v_i})\}_{i=1}^{n_C}$, where $v_i$ is a node, $t^C_{v_i}$ is its activation time in $C$, $|C| = n_C$ is a size of a cascade. Note that we do not observe who infected whom.

*Communities.* We assume that $G$ is partitioned into communities: $\mathcal{A} = \{A_1, \dots, A_k\}$, $\cup_{i=1}^k A_i = V$, $A_i \cap A_j = \emptyset$ for $i \neq j$. We expect to observe a high intra-community density of edges compared to inter-community density. In our experiments, the ground truth partitions $\mathcal{A}$ are available for all datasets (see Section 5.1.1). By observing only a set of cascades $\mathbf{C}$ we want to find a partition $\mathcal{A}'$ similar to $\mathcal{A}$.

### 3.2 Cascade Models

*3.2.1 SIR model.* The main model for our experiments is a well-known SIR (Susceptible-Infected-Recovered) model [15]. Each node in the network can be in one of the three states: susceptible, infected, or recovered. An infected node infects its susceptible neighbors with rate $\alpha$, the infection is spread simultaneously and independently along all edges. An infected node recovers with rate $\beta$ and then stops spreading infection in the network. For each cascade $C$ we sample its own infection rate $\alpha_C$ from the Lomax (shifted Pareto) distribution in order to model a variety of cascades: there can be minor or widely circulated news, small scale epidemics or pandemics, etc. The source node of a cascade is chosen uniformly at random.

*3.2.2 SI model with bounded duration (SI-BD).* In some cases, SIR model might not be tractable for theoretical analysis, so we assume a simpler diffusion model introduced in [28]. In this model, again, an activated node infects its neighbors after an exponentially distributed time with intensity $\alpha$, but there is no recovery rate. Instead, all nodes recover simultaneously at some threshold time $T_{max}$ and the epidemic stops. For simplicity we further assume $T_{max}$ to be fixed, but our methods allow varying $T_{max}$ for different epidemics.

*3.2.3 Community-based SI-BD model (C-SI-BD).* Another model which allows for a simpler theoretical analysis is based on the setting from [33]. It is assumed that the spreading does not occur over the edges of a graph $G$ but depends solely on the community structure $\mathcal{A}$. As before, the first node of a cascade is chosen uniformly at random. Each activated node can infect all other susceptible nodes independently after an exponentially distributed time. If a susceptible node belongs to the same community, then the infection rate is $\alpha_{in}$, otherwise it is $\alpha_{out}$, $\alpha_{out} < \alpha_{in}$. Epidemic stops at time $T_{max}$.

## 4 ALGORITHMS

### 4.1 Background on Likelihood Maximization

Recall that $t_i^C$ denotes the activation time for node $i$ in cascade $C$; we often omit the index $C$ when the context is clear. Without loss of generality, for the first node of a cascade, we set $t_i = 0$. Finally, if a node $i$ is not infected during an epidemic, then we set $t_i = T_{max}$.

Denote $\Delta^C_{i,j} = \Delta_{i,j} = |t_i - t_j|$. The log-likelihood $\log L(C)$ of the cascade $C$ for SI-BD with varying infection rates ($i$ infects a susceptible neighbor $j$ after an exponentially distributed time with rate $\alpha_{i,j}$) is given in [28]. For our purposes, it is convenient to write this expression as:

$$\log L(C) = -\sum_{i,j:i<j} \alpha_{i,j}\Delta_{i,j} + \sum_{i:t_i\in(0,T_{max})} \log \sum_{j:t_j<t_i} \alpha_{j,i}\,. \quad (1)$$

The log-likelihood for all cascades is $\log L(\mathbf{C}) = \sum_{C\in\mathbf{C}} \log L(C)$. We will next introduce two methods based on maximizing $\log L(\mathbf{C})$.

### 4.2 ClustOpt

Consider the C-SI-BD model described in Section 3.2.3. Denote by $a(i)$ the community assignment for a node $i$. Then C-SI-BD is equivalent to the model used in [28] with $\alpha_{i,j} = \alpha_{in}$ if $a(i) = a(j)$ and $\alpha_{i,j} = \alpha_{out}$ otherwise. The log-likelihood in (1) becomes:

$$\log L(C, \mathcal{A}) = -(\alpha_{in} - \alpha_{out}) \sum_{\substack{i,j:i<j,\\ a(i)=a(j)}} \Delta_{i,j} - \alpha_{out} \sum_{i,j:i<j} \Delta_{i,j}$$
$$+ \sum_{i:t_i\in(0,T_{max})} \log \Big( (\alpha_{in} - \alpha_{out})|\{j : t_j < t_i, a(j) = a(i)\}|$$
$$+\alpha_{out}|\{j : t_j < t_i\}| \Big)\,. \quad (2)$$

.

We propose the following algorithm to maximize (2).

**ALGORITHM 1:** ClustOpt
```
(1) Find initial partition $\mathcal{A}_{init}$;
(2) Find $\hat{\alpha}_{in}, \hat{\alpha}_{out} = \arg\max_{\alpha_{in},\alpha_{out}} \log L(\mathbf{C}, \mathcal{A}_{init})$;
(3) For fixed $\hat{\alpha}_{in}, \hat{\alpha}_{out}$ find $\hat{\mathcal{A}} = \arg\max_{\mathcal{A}} \log L(\mathbf{C}, \mathcal{A})$.
```

Let us now discuss how the steps (1)–(3) are implemented.

*Step (1):* We have noticed that a stable and fast approach is to start from some initial reasonable partition and optimize over $\alpha_{in}$, $\alpha_{out}$ and $\mathcal{A}$ only once, thus avoiding iterations of the costly optimization steps (2) and (3). In the current paper we propose to start from Clique(0) explained in Section 4.4.

*Step (2):* Without loss of generality, we set $\alpha_{in} = (\delta + 1)\alpha_{out}$. Substituting this into (2), we find optimal $\alpha_{out}$ in terms of $\delta$ and $\mathcal{A}$:

$$\hat{\alpha}_{out} = \frac{\sum_{C \in \mathcal{C}} (|C| - 1)}{\sum_{C \in \mathcal{C}} \left( \delta \sum_{i,j:a(i)=a(j)} \Delta^C_{i,j} + \sum_{i,j} \Delta^C_{i,j} \right)}. \quad (3)$$

Unfortunately, due to the summation of logarithms in (2), we cannot find another simple analytical relation between the optimal values of $\delta$ and $\alpha_{out}$. Hence, we resort to a numerical solution. Due to (3), it is sufficient to numerically find only the optimal $\delta$, this will give us the optimal values for both $\alpha_{in}$ and $\alpha_{out}$.

*Step (3):* We follow [25], where the Louvain algorithm [6] is modified for optimizing a wide range of functions that depend on community structure, besides the original modularity measure. We adapt this algorithm for the likelihood given in (2) by computing the gain in $\log L(\mathcal{C}, \mathcal{A})$ obtained by moving a node $v$ from one community to another. In view of computational complexity, we consider only moving single nodes from one community to another and do not attempt to move groups of nodes or merge communities.

## 4.3 GraphOpt

In this section, we drop the assumption of fixed infection rates within and between communities. Instead, we assume the SI-BD cascade model defined in Section 3.2.2. Our task is much more complex in this setting because now we have a hidden graph $G$.

We propose an expectation-maximization-based method, where $G$ is a latent variable. We assume the following generative probabilistic process. For a given partition $\mathcal{A}$ of $n$ nodes we construct a graph $G$ according to some model (distribution) $\mathrm{P}(G|\mathcal{A})$. Then, based on $G$, we generate a set of cascades $\mathcal{C}$ according to another model $\mathrm{P}(\mathcal{C}|G)$. We observe $\mathcal{C}$ and our aim is to recover the partition $\bar{\mathcal{A}}$ which maximizes the likelihood, i.e., $\bar{\mathcal{A}} = \arg\max_{\mathcal{A}} \mathrm{P}(\mathcal{C}|\mathcal{A})$. Let us first explain how the standard expectation-maximization (EM) approach applies to this problem:

(1) Choose some initial partition $\hat{\mathcal{A}}$;
(2) Find the distribution $\mathrm{P}(G|\mathcal{C}, \hat{\mathcal{A}})$;
(3) Update $\hat{\mathcal{A}}$: $\hat{\mathcal{A}} = \arg\max_{\mathcal{A}} \mathbb{E}_G \log \mathrm{P}(\mathcal{A}\, G|\mathcal{C})$;
(4) Iterate (2)-(3) until convergence.

This algorithm cannot be applied directly because, as we explain below, steps (2) and (3) are computationally intractable. Our algorithm GraphOpt is obtained by simplifying these two steps.

First, let us rewrite the distribution in (2):

$$\mathrm{P}(G|\mathcal{C}, \hat{\mathcal{A}}) = \frac{\mathrm{P}(\mathcal{C}\, G\, \hat{\mathcal{A}})}{\mathrm{P}(\mathcal{C}\, \hat{\mathcal{A}})} = \frac{\mathrm{P}(\mathcal{C}|G, \hat{\mathcal{A}})\mathrm{P}(G\, \hat{\mathcal{A}})}{\mathrm{P}(\mathcal{C}|\hat{\mathcal{A}})\mathrm{P}(\hat{\mathcal{A}})}$$
$$= \frac{\mathrm{P}(\mathcal{C}|G)\mathrm{P}(G|\hat{\mathcal{A}})}{\mathrm{P}(\mathcal{C}|\hat{\mathcal{A}})} \propto \mathrm{P}(\mathcal{C}|G)\mathrm{P}(G|\hat{\mathcal{A}}), \quad (4)$$

where we used that $\mathrm{P}(\mathcal{C}|\hat{\mathcal{A}})$ does not depend on $G$ and $\mathrm{P}(\mathcal{C}|G, \hat{\mathcal{A}}) = \mathrm{P}(\mathcal{C}|G)$ since, given the graph, the division into communities does not affect the cascades.

Now, we rewrite the probability in (3): $\mathrm{P}(\mathcal{A}\, G|\mathcal{C}) = \frac{\mathrm{P}(\mathcal{C}\, G\, \mathcal{A})}{\mathrm{P}(\mathcal{C})} = \frac{\mathrm{P}(\mathcal{C}|G)\mathrm{P}(G|\mathcal{A})\mathrm{P}(\mathcal{A})}{\mathrm{P}(\mathcal{C})}$. Since we do not have any prior assumptions for $\mathcal{A}$, we assume that $\mathrm{P}(\mathcal{A})$ is constant. Also, $\mathrm{P}(\mathcal{C})$ and $\mathrm{P}(\mathcal{C}|G)$ do not depend on $\mathcal{A}$, so

$$\arg\max_{\mathcal{A}} \mathbb{E}_G \log \mathrm{P}(\mathcal{A}\, G|\mathcal{C}) = \arg\max_{\mathcal{A}} \mathbb{E}_G \log \mathrm{P}(G|\mathcal{A}). \quad (5)$$

From (4) and (5) we see the computational bottlenecks of the EM algorithm. Indeed, even if for each $G$ we can estimate the probability in (4), it is still impossible to even compute $\mathbb{E}_G \log \mathrm{P}(G|\mathcal{A})$, because the expectation is over all possible realizations of $G$. Clearly, we cannot hope to maximize this expression over $\mathcal{A}$. Therefore, we simplify the procedure: instead of computing the entire probability distribution over all graphs $G$ in (4), we find only the most likely graph $\hat{G}$, with the idea that $\hat{G}$ gives the largest contribution to $\mathbb{E}_G \log \mathrm{P}(G|\mathcal{A})$. Then the optimization problem in (5) is solved only for $\hat{G}$. As a result, we obtain the following algorithm.

**ALGORITHM 2:** GraphOpt

```
(1) Choose some initial partition $\hat{\mathcal{A}}$ and graph $\hat{G}$;
(2) Update $\hat{G}$: $\hat{G} = \arg\max_G \left(\log \mathrm{P}(\mathcal{C}|G) + \log \mathrm{P}(G|\hat{\mathcal{A}})\right)$;
(3) Update $\hat{\mathcal{A}}$: $\hat{\mathcal{A}} = \arg\max_{\mathcal{A}} \log \mathrm{P}(\hat{G}|\mathcal{A})$;
(4) Iterate (2)-(3) until convergence.
```

Let us describe how each step of Algorithm 2 is implemented.

*Step (1):* We start from a trivial partition $\hat{\mathcal{A}}$ in which each node belongs to its own cluster. We also create an initial graph[1] $\hat{G} = (V, E_0)$ as follows: for each cascade $C$ consider the path of $|C| - 1$ edges, where each edge connects two nodes that are subsequently activated in this cascade; $E_0$ is a union of such paths over all $C \in \mathcal{C}$.

*Step (2):* $\mathrm{P}(G|\hat{\mathcal{A}})$ is defined by the ILFR model proposed in [25], where the formula for $\log \mathrm{P}(G|\hat{\mathcal{A}})$ is also provided. This model was shown to give the best fit to a variety of real-world networks in terms of likelihood. For $\mathrm{P}(\mathcal{C}|G)$, we assume the SI-BD cascade model, so we use (1) with $\alpha_{i,j} = \alpha$ if $i$ and $j$ are connected in $G$ and $\alpha_{i,j} = 0$ otherwise. Then $\log \mathrm{P}(\mathcal{C}|G) = \sum_{C \in \mathcal{C}} \log \mathrm{P}(C|G)$ with

$$\log \mathrm{P}(C|G) = -\alpha \sum_{(i,j) \in E(G)} \Delta_{i,j} + (|C| - 1) \log \alpha$$
$$+ \sum_{i: t_i \in (0, T_{max})} \log |\{j : t_j < t_i, (i,j) \in E(G)\}|. \quad (6)$$

A nice feature of (6) is that for given $\mathcal{C}$ and $G$ it is easy to compute $\alpha$ that maximizes the overall likelihood:

$$\alpha_{opt} = \frac{\sum_{C \in \mathcal{C}} (|C| - 1)}{\sum_{C \in \mathcal{C}} \sum_{(i,j) \in E(G)} \Delta^C_{i,j}}. \quad (7)$$

Finally, we need to find $\hat{G} = \arg\max_G \left(\log \mathrm{P}(G|\hat{\mathcal{A}}) + \log \mathrm{P}(\mathcal{C}|G)\right)$. We use a greedy approach to approximately find $\hat{G}$: we iteratively add and remove edges to increase $\log \mathrm{P}(G|\hat{\mathcal{A}}) + \log \mathrm{P}(\mathcal{C}|G)$ (the detailed description is omitted due to space constrains).

*Step (3):* This is a standard likelihood optimization task used in community detection. We use the Louvain-based algorithm proposed in [25].

[1]This initial graph is needed for step (2), since our update algorithm requires initialization with non-zero likelihood.

## 4.4 Clustering of Surrogate Graphs

In this section, we present simple yet effective methods that construct a surrogate graph $\hat{G}$ and then cluster this graph (using the Louvain algorithm [6]). It is crucial that $\hat{G}$ does *not* need to be similar to $G$, it just needs to capture the community structure on an aggregated level. Our experiments show that clustering of $\hat{G}$ often performs better than first inferring $G$ and then clustering it.

*4.4.1 Path algorithm.* Assume again that cascades $\mathcal{C}$ are generated by the SI-BD model. Now, for $C \in \mathcal{C}$ let $G' = G'(C)$ be a graph with exactly $|C| - 1$ edges that maximizes the probability $\mathrm{P}(C|G')$ in (6). It is easy to check that $G'$ is merely a path connecting subsequently activated nodes. Note that we have already used such paths to initialize Algorithm 2. This leads to the following algorithm.

```
ALGORITHM 3: Path
    (1) Construct weighted graph $\hat{G}$, where the weight of each
        edge $e$ in $\hat{G}$ is the number of $G'(C)$ including $e$;[2]
    (2) Find clusters in $\hat{G}$ using the Louvain algorithm.
```

*4.4.2 Clique.* Another approach is to include *all* possible edges that could participate in the cascade weighing them by the proxy of their likelihood. Then each cascade $C$ results in a weighted clique of size $|C|$. The weights can be chosen, for example, as follows. For a cascade $C$ and two nodes $i, j$ let us consider the probability $P^C(i,j) = \mathrm{P}(j \text{ was infected from } i|C)$. If $t_i > t_j$, then, obviously, $P^C(i,j) = 0$. If $t_i < t_j$, then, as in [11], we assume that $P^C(i,j)$ decreases exponentially with $\Delta_{i,j}$; namely, $P^C(i,j) = c_j\, e^{-a\Delta_{i,j}}$, where $c_j$ is a constant depending on $j$. Since $j$ was infected from exactly one previous node, we must have $\sum_{i:t_i<t_j} P^C(i,j) = 1$. Therefore, $P^C(i,j) = \frac{e^{-a\Delta_{i,j}}}{\sum_{l:t_l<t_j} e^{-a\Delta_{l,j}}}$.

Clique constructs $\hat{G}$ as the weighted graph with weight of $\{i,j\}$ given by $\sum_{C\in\mathcal{C}}(P^C(i,j) + P^C(j,i))$, which, under our assumptions, is the expected number of times infections passed between $i$ and $j$. Note that we directly model $P^C(i,j)$ rather than making assumptions about infection times. Parameter $a$ essentially balances between paths and cliques: if $a$ is large, then we mostly take into account subsequent nodes with small $\Delta_{i,j}$; for small $a$ all pairs of nodes participated in $C$ are important. To make Clique insensitive to the speed of epidemics, we take $a = \frac{1}{\Delta}$, where $\Delta$ is the average time between infected times (we average over all pairs of infected nodes belonging to the same cascade). We also consider Clique(0) with $a = 0$, which is a natural choice for the SI-BD model because it mimics the memory-less property of exponential infection times.[3]

```
ALGORITHM 4: Clique
    (1) Construct $\hat{G}$ as the weighted graph with weight of $\{i,j\}$
        given by $\sum_{C\in\mathcal{C}}(P^C(i,j) + P^C(j,i))$.
    (2) Find clusters in $\hat{G}$ using the Louvain algorithm.
```

[2]Weighting edges is important, since for large $|\mathcal{C}|$ unweighted union of edges can be a complete graph, with no meaningful partition.

[3]We noticed that Clique is not too sensitive to varying $a$ in a reasonable interval, while for too large $a$ Clique becomes very similar to Path.

## 4.5 Baselines

MultiTree uses the algorithm from [30] to find an inferred graph $\hat{G}$, which is then clustered by the Louvain algorithm [6].

Oracle algorithm can be considered as superior for all possible network inference algorithms: we construct a graph $\hat{G}$ consisting of all edges participated in cascades (assuming these edges are known, hence, the name Oracle). The obtained graph $\hat{G}$ is clustered by the Louvain algorithm.

Finally, we used algorithms proposed for solving the same problem as in our work: C-IC and C-Rate [5], as well as R-CoDi and D-CoDi [26]. We use the publicly available implementations provided by the authors. For C-IC and C-Rate we use the real number of communities as an input parameter.[4]

## 5 EXPERIMENTS

## 5.1 Experimental Setup

*5.1.1 Networks.* We used real-world datasets of different sizes and nature, for which the ground truth partition of nodes into $k$ non-overlapping communities is available: Zachary's karate club [39] ($n = 34$, $m = 78$, $k = 2$), dolphin social network [20] ($n = 62$, $m = 159$, $k = 2$), American college football [24] ($n = 115$, $m = 613$, $k = 11$), books about politics [23] ($n = 105$, $m = 441$, $k = 3$), political blogs [1] ($n = 1224$, $m = 16715$, $k = 2$), email-Eu-core dataset [19] ($n = 986$, $m = 16064$, $k = 42$).

*5.1.2 Metric.* We evaluate the quality of the obtained communities compared to the ground truth using the Normalized Mutual Information (NMI) similarity measure [2, 9]. Let $V_0 \subset V$ denote the set of all nodes that participated in cascades. To compute NMI, we assign all nodes in $V \backslash V_0$ to one cluster labeled "unknown".

*5.1.3 Cascades.* We took the datasets with ground truth communities described in Section 5.1.1 and, in order to cover various possible diffusion processes, we generated the cascades according to all models discussed in Section 3.2. Importantly, there are both edge-based and community-based models.

W.l.o.g., we set $\beta = 1$ for the SIR model and $T_{max} = 1$ for SI-BD and C-SI-BD. We noticed that to get an informative set of cascades, the parameters $\alpha$, $\alpha_{in}$, $\alpha_{out}$, and the parameter of the Lomax distribution used by SIR have to be different for different datasets. For SIR and SI-BD we choose parameters so that the average size of a cascade is 2; for C-SI-BD we take $\alpha_{in} = 10\alpha_{out}$ and choose $\alpha_{out}$ such that the number of cascades consisting of one node is about 20%. We checked that the obtained distribution of cascade sizes is similar to observed for real data [3, 10, 12, 17, 18]. Single-node cascades were removed.

## 5.2 Results

We have run the algorithms on all datasets and varied the number of cascades $|\mathcal{C}|$. Figures 1 and 2 show the results obtained for SIR and C-SI-BD models, respectively.[5] All results are averaged over 5 samples of generated cascades $\mathcal{C}$.

[4]Although there are modifications of these algorithms automatically choosing the number of communities, these modifications often failed on our data.

[5]Due to space constraints, we omit plots for the SI-BD model since they lead to similar conclusions.

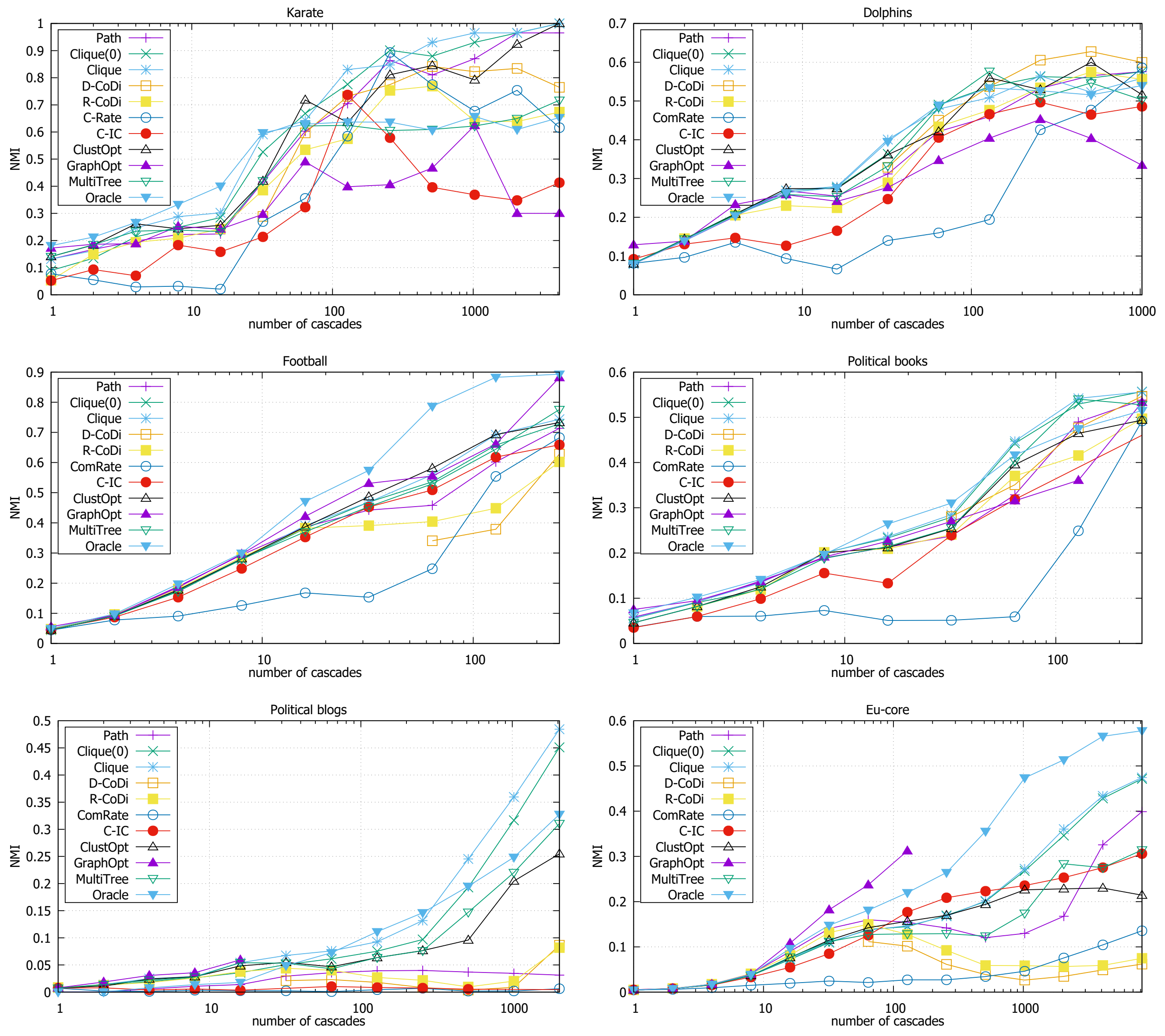

**Figure 1: Comparison of algorithms on real-world datasets, SIR cascades**

We noticed that the results are quite different for different datasets and cascades models. Nevertheless, the proposed Clique algorithm has the most stable performance, implying that the corresponding surrogate graph is able to capture the information about the community structure. Importantly, both Clique(0) and Clique work equally well for all cascade models since they are not based on a particular one, as discussed in Section 4.4.

Note that Path is usually worse than Clique as it uses less information.

The baselines C-IC and C-Rate are very unstable and in many cases they show the worst results. This is expected, since C-IC and C-Rate are based on a specific community-based model. Indeed, on C-SI-BD C-IC shows a reasonable performance in some cases, especially on Political blogs. Note that C-IC is almost always better than C-Rate (except for some cascade sizes on Karate with SIR model). D-CoDi and R-CoDi are essentially based on clustering of a surrogate graph, so their performance is stable for all epidemic models. However, in all cases D-CoDi failed on cascades of small sizes (we included only successful points). D-CoDi can be both better and worse than R-CoDi. Having reasonable performance in many cases, these two baselines may have unexpectedly bad points: see, e.g., the large number of cascades on Political blogs and Eu-Core with all epidemic models. MultiTree turns out to have

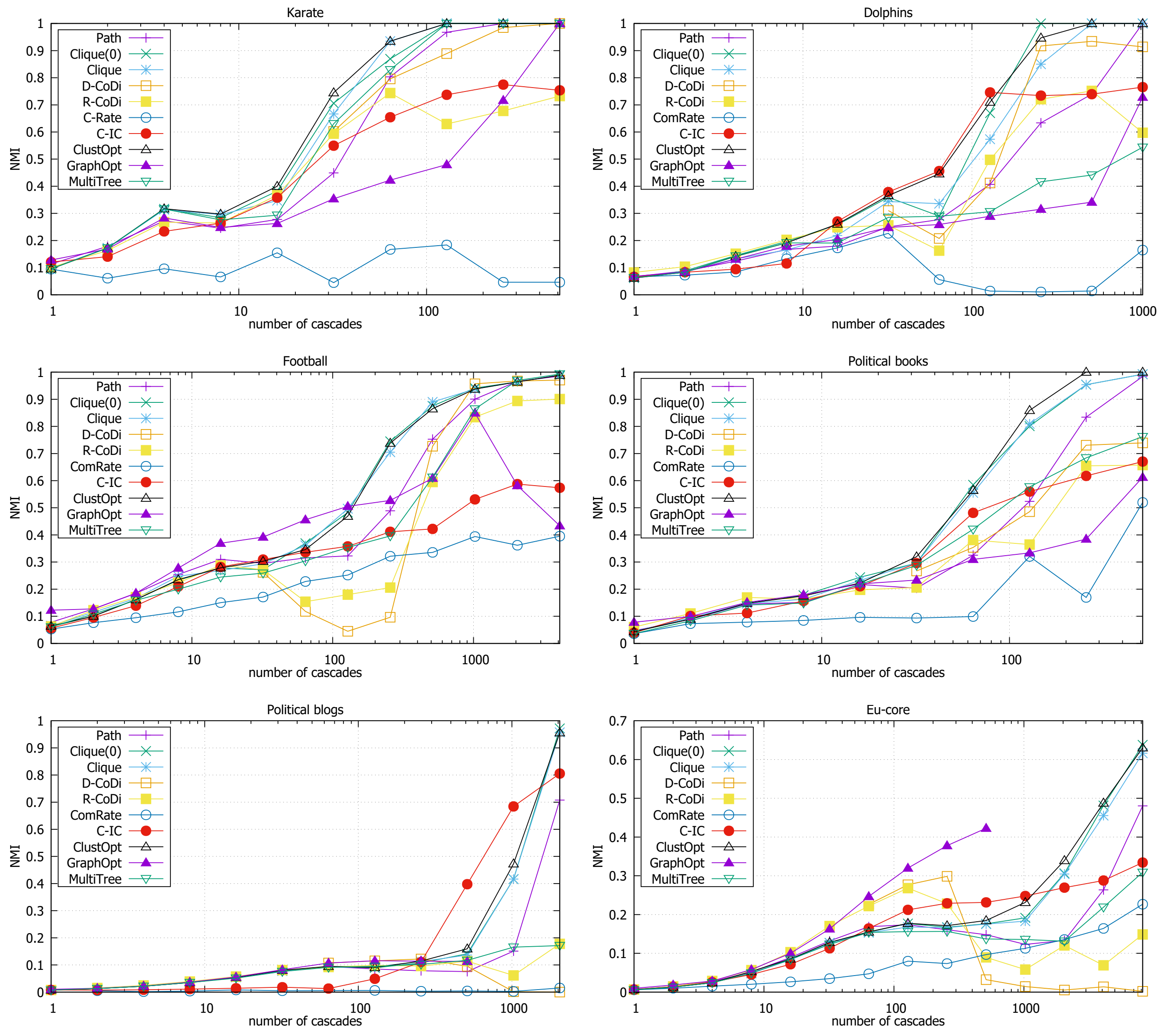

**Figure 2: Comparison of algorithms on real-world datasets, C-SI-BD cascades**

the most stable performance among the baselines but it is usually beaten by Clique(0) and Clique.

GraphOpt shows an excellent performance, e.g., on Eu-core for all types of epidemics, on Football for SIR cascades and for small number of C-SI-BD cascades. However, in general its quality is unstable, and it is also much slower than surrogate-based methods.

Oracle is considered only as an upper bound for all network inference algorithms.[6] If the number of cascades is large enough, then Oracle essentially clusters the original graph. Interestingly, in many cases Oracle is beaten by the surrogate-graph-based algorithms. This basically means that errors made by Louvain on original graph can be reduced by clever weighting of edges provided by the surrogate-graph-based algorithms.

To conclude, we propose using the universal Clique method for the problem of community inference based on cascade data. This algorithm is simple, efficient, and works well for all considered cascade models.

## ACKNOWLEDGMENTS

This study was funded by RFBR according to the research project 18-31-00207.

[6]We do not plot Oracle for C-SI-BD, since in this model graph is not used by the propagation processes and. In particular, for large $|C|$ we may get a complete graph.

## REFERENCES

[1] Lada A Adamic and Natalie Glance. 2005. The political blogosphere and the 2004 US election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*. ACM, 36–43.

[2] James P Bagrow. 2008. Evaluating local community methods in networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 05 (2008), P05001.

[3] Eytan Bakshy, Jake M Hofman, Winter A Mason, and Duncan J Watts. 2011. Identifying influencers on twitter. In *Fourth ACM International Conference on Web Seach and Data Mining (WSDM)*.

[4] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. 2013. Influence-based network-oblivious community detection. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 955–960.

[5] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. 2017. Efficient methods for influence-based network-oblivious community detection. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 2 (2017), 32.

[6] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[7] Hadi Daneshmand, Manuel Gomez-Rodriguez, Le Song, and Bernhard Schoelkopf. 2014. Estimating Diffusion Network Structures: Recovery Conditions, Sample Complexity & Soft-thresholding Algorithm.. In *ICML*. 793–801.

[8] Nan Du, Le Song, Ming Yuan, and Alex J Smola. 2012. Learning networks of heterogeneous influence. In *Advances in Neural Information Processing Systems*. 2780–2788.

[9] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3 (2010), 75–174.

[10] Wojciech Galuba, Karl Aberer, Dipanjan Chakraborty, Zoran Despotovic, and Wolfgang Kellerer. 2010. Outtweeting the twitterers-predicting information cascades in microblogs. *WOSN* 10 (2010), 3–11.

[11] Manuel Gomez Rodriguez, Jure Leskovec, and Andreas Krause. 2010. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1019–1028.

[12] Manuel Gomez-Rodriguez, Jure Leskovec, and Bernhard Schölkopf. 2013. Modeling Information Propagation with Survival Theory.. In *ICML*. 666–674.

[13] Manuel Gomez Rodriguez, Jure Leskovec, and Bernhard Schölkopf. 2013. Structure and dynamics of information pathways in online media. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 23–32.

[14] Lars Hufnagel, Dirk Brockmann, and Theo Geisel. 2004. Forecast and control of epidemics in a globalized world. *Proceedings of the National Academy of Sciences of the United States of America* 101, 42 (2004), 15124–15129.

[15] Matt J Keeling and Ken TD Eames. 2005. Networks and epidemic models. *Journal of the Royal Society Interface* 2, 4 (2005), 295–307.

[16] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 137–146.

[17] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a social network or a news media?. In *Proceedings of the 19th international conference on World wide web*. AcM, 591–600.

[18] Kristina Lerman, Rumi Ghosh, and Tawan Surachawala. 2012. Social contagion: An empirical study of information spread on Digg and Twitter follower graphs. *arXiv preprint arXiv:1202.3162* (2012).

[19] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2.

[20] David Lusseau, Karsten Schneider, Oliver J Boisseau, Patti Haase, Elisabeth Slooten, and Steve M Dawson. 2003. The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology* 54, 4 (2003), 396–405.

[21] Seth Myers and Jure Leskovec. 2010. On the convexity of latent social network inference. In *Advances in Neural Information Processing Systems*. 1741–1749.

[22] Praneeth Netrapalli and Sujay Sanghavi. 2012. Learning the graph of epidemic cascades. *ACM SIGMETRICS Performance Evaluation Review* 40, 1 (2012), 211–222.

[23] Mark EJ Newman. 2006. Modularity and community structure in networks. *Proceedings of the national academy of sciences* 103, 23 (2006), 8577–8582.

[24] Mark EJ Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical review E* 69, 2 (2004), 026113.

[25] Liudmila Prokhorenkova and Alexey Tikhonov. 2019. Community detection through likelihood optimization: in search of a sound model. *Proceedings of the 2019 World Wide Web Conference (WWW '19)* (2019).

[26] Maryam Ramezani, Ali Khodadadi, and Hamid R Rabiee. 2018. Community Detection Using Diffusion Information. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 12, 2 (2018), 20.

[27] Maryam Ramezani, Hamid R Rabiee, Maryam Tahani, and Arezoo Rajabi. 2017. DANI: A Fast Diffusion Aware Network Inference Algorithm. *arXiv preprint arXiv:1706.00941* (2017).

[28] Manuel Gomez Rodriguez, David Balduzzi, and Bernhard Schölkopf. 2011. Uncovering the temporal dynamics of diffusion networks. *arXiv preprint arXiv:1105.0697* (2011).

[29] Manuel Gomez Rodriguez, Jure Leskovec, David Balduzzi, and Bernhard Schölkopf. 2014. Uncovering the structure and temporal dynamics of information propagation. *Network Science* 2, 01 (2014), 26–65.

[30] Manuel Gomez Rodriguez and Bernhard Schölkopf. 2012. Submodular inference of diffusion networks from multiple trees. *arXiv preprint arXiv:1205.1671* (2012).

[31] Daniel M Romero, Brendan Meeder, and Jon Kleinberg. 2011. Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter. In *Proceedings of the 20th international conference on World wide web*. ACM, 695–704.

[32] Kazumi Saito, Masahiro Kimura, Kouzou Ohara, and Hiroshi Motoda. 2009. Learning continuous-time information diffusion model for social behavioral data analysis. In *Asian Conference on Machine Learning*. Springer, 322–337.

[33] Jaron Sanders and Alexandre Proutière. 2017. Optimal Clustering Algorithms in Block Markov Chains. *arXiv preprint arXiv:1712.09232* (2017).

[34] Arlei Silva, Hérico Valiati, Sara Guimarães, and Wagner Meira Jr. 2011. From individual behavior to influence networks: A case study on twitter. In *Proc. of the 17th Brazilian Symposium on Multimedia, Hypermedia and Web*.

[35] Tristan Mark Snowsill, Nick Fyson, Tijl De Bie, and Nello Cristianini. 2011. Refining causality: who copied from whom?. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 466–474.

[36] Chenxi Wang, John C Knight, and Matthew C Elder. 2000. On computer viral infection and the effect of immunization. In *Computer Security Applications, 2000. ACSAC'00. 16th Annual Conference*. IEEE, 246–256.

[37] Liaoruo Wang, Stefano Ermon, and John E Hopcroft. 2012. Feature-enhanced probabilistic models for diffusion network inference. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 499–514.

[38] Shuang-Hong Yang and Hongyuan Zha. 2013. Mixture of mutually exciting processes for viral diffusion. In *International Conference on Machine Learning*. 1–9.

[39] Wayne W Zachary. 1977. An information flow model for conflict and fission in small groups. *Journal of anthropological research* 33, 4 (1977), 452–473.

[40] Ke Zhou, Hongyuan Zha, and Le Song. 2013. Learning Social Infectivity in Sparse Low-rank Networks Using Multi-dimensional Hawkes Processes.. In *AISTATS*, Vol. 13. 641–649.