Temporal Ordered Clustering in Dynamic Networks: Unsupervised and Semi-supervised Learning Algorithms

Anonymous Author(s)*

ABSTRACT

In *temporal ordered clustering*, given a single snapshot of a dynamic network, in which nodes arrive at distinct time instants along with edges, we aim at partitioning its nodes into K ordered clusters $C_1 \prec \cdots \prec C_K$ such that for $i < j$, nodes in cluster C_i arrived before nodes in cluster C_j, with K being a data-driven parameter and not known upfront. Such a problem of inferring the evolution of a dynamic network is of considerable significance in many applications in social networks ranging from tracking the expansion of fake news to mapping the spread of information and thereby designing personalized channels by prioritizing information of shared interest. We first formulate our problem for a general dynamic graph, and propose an integer programming framework that finds the optimal clustering scheme, equivalently represented as a strict partial order set, achieving the best precision (i.e., fraction of successfully ordered node pairs) for a fixed density (i.e., fraction of comparable node pairs). Guided by the linear programming approximation of the optimization problem with coefficients estimated via sequential importance sampling, we design unsupervised and semi-supervised algorithms to find temporal ordered clusters. To illustrate our techniques, we also apply our methods to a network model – the vertex copying model or the duplication-divergence model – which turns out to present a real challenge when compared to other network models. Finally, we validate the proposed algorithms on synthetic data and various real-world networks against upper bounds given by linear programming and simple greedy heuristics.

KEYWORDS

Clustering, dynamic networks, supervised learning, semi-supervised learning, temporal order

1 INTRODUCTION

The clustering of nodes is a classic problem in networks. In its typical form in static networks, it finds communities where methods like spectral clustering, modularity maximization, minimum-cut method, and hierarchical clustering are commonly used [13].

However, in dynamic networks that are growing over time, criteria of clustering based on the temporal characteristics of networks find significant relevance in practice. One such approach is guided by the problem of node labeling according to their arrival order when only the structure of a final snapshot of a dynamic network is provided. The availability of only structure means that either we are given an unlabeled graph or the current node labels do not provide any useful information. As it turns out, in many real-world networks and graph models, it is impossible to find a complete order of arrival of nodes due to a large number of symmetries inherent in the graph. Figure 1 shows an example. In such cases, it is quintessential to classify nodes that are indistinguishable themselves in terms of arrival order into a cluster. Furthermore, the formed clusters $\{C_i\}$ also needs to be ordered as $C_1 \prec C_2 \prec \cdots$ so that in the input snapshot of the dynamic network, for any $i < j$, all the nodes in the cluster C_i are estimated to be arrived earlier than all the nodes in the cluster C_j, and all the nodes inside each cluster are considered to be identical in arrival order. We call such a clustering scheme as *temporal ordered clustering*.

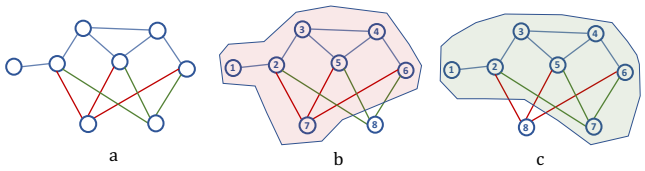


Figure 1: Example showing how temporal clustering arises: a) the input graph without labels. b) and c) arbitrary labelings with the bottom two nodes as the last two arrived, without any graph model assumption. Note that in Fig. b and c, nodes 7 and 8 have the same set of neighbors. If we simulate the process of evolution, we observe that graphs in Fig b and c at time 7 (i.e., with nodes $1-7$) are identical. Thus in Fig b and c, we cannot distinguish between the nodes 8 and 7 as to which arrived last.

Temporal ordered clustering is related to many applications in practice. For example in online social networks, it can be useful to disseminate specific information or advertisements targeted at nodes that arrived around the same time. In biological networks, it identifies the evolution of biomolecules in the network and helps in predicting early proteins that are known to be preferentially implicated in cancers and other diseases [16]. In rumor or epidemic networks, temporal ordered clustering can assist in identifying the sources and carriers of false information.

Our contributions. In this paper, we present the following key results:

- We provide a general framework for solving temporal ordered clustering in dynamic networks when only the final snapshot of its evolution is provided. Later, with the knowledge of the probabilistic evolution of the graph model, we derive an optimization problem for temporal ordered clustering. The optimization problem depends on, for any node pair (u, v), the probability that node u is older than node v, denoted by p_{uv}. We design a Markov chain based sequential importance sampling algorithm to estimate p_{uv}, and prove its convergence. The solution to a linear programming relaxation of the original optimization problem, with coefficients as estimated p_{uv}, presents an upper bound on the clustering quality.
- In the case of large networks, when the complexity for solving the original optimization is higher, approximate solutions, which directly make use of the estimated p_{uv}, are developed. Moreover, when the real networks may not fit well within the probabilistic graph model under consideration, we develop semi-supervised

techniques that exploit the graph structure to improve estimation precision. We observe that the use of semi-supervised learning enhances the estimated values of p_{uv} quickly even with a small percentage of labeled data.

- In the final sections of the paper, as an application of the proposed general technique, we focus on duplication-divergence or vertex copying dynamic network model (DD-model) in which, informally, a new node copies the edges of a randomly selected existing node and retains them with a certain probability, and also makes random connections the remaining nodes (see Section 5.1 for details). The DD-model poses unique challenges in comparison with other graph models, because of the features listed below:
  - *Non-equiprobable permutations*: In many of the graph models including the preferential attachment and Erdős-Rényi graph models, all the feasible permutations representing node arrival order are equally likely [10]. Later in this paper, we show that this is not the case in DD-model.
  - *Large number of symmetry*: We provide evidence of a large number of automorphisms in a duplication-divergence graph, where as Erdős-Rényi and preferential attachment graphs are asymmetric with high probability.
  - *Ineffectiveness of degree-based techniques*: In some models (including preferential attachment model), the oldest nodes have larger expected degrees than the youngest nodes over time, with high probability. But in the DD-model, we show that the average degree does not exhibit such a consistent trend. Thus any method based on degrees is bound to fail in the DD-model.

Prior related work. Graph clustering is a well-studied problem which, in general, follows two main approaches: 1) define a similarity metric between node pairs, and choose clusters so as to maximize similarity among the nodes inside a cluster and minimize similarity between nodes in different clusters; 2) identify subgraphs within the input graph that reach a certain value of fitness measure, usually based on subgraph density, conductance, normalized cut or sparse cut [13]. Many of the clustering techniques on static graphs have been extended to dynamic graphs, where primarily the aim was to study the evolution of fitness or similarity based clusters [3, 4, 8, 9].

The temporal ordered clustering considered in this paper poses a very different problem in contrast to the classical formulation. The optimization criterion for temporal ordered clustering introduces a fresh look taking into account the graph model and its temporal behavior (see Section 3). Incidentally, the main aim of our clustering is to characterize the inability of recovering the history of a dynamic network due to the symmetries exhibited by the associated model. The nodes inside our clusterings are indistinguishable in terms of their arrival order due to symmetries in the input graph and there exists a hierarchy or order among the clusters with respect to graph evolution.

Previous works on semi-supervised clustering methods for data represented as vectors [1, 2] and their extensions to graphs [6] focus mainly on using the labeled nodes to predefine clusters and their centroids. However, in temporal ordered clustering, the labeled nodes need not fully represent all the clusters, and they are used to reduce the complexity of estimation of coefficients of the associated linear programming (by restricting the sampling distribution of importance sampling, see Section 4.2)

Node arrival order in the DD-model has been studied in [7] and [11], and the references therein. Most of the prior works focus on getting the complete arrival order of nodes (total order), but it turns out that it becomes nearly impossible due to the inherent symmetries. Instead of total order, in this work we focus on deriving an optimal partial order of nodes which is equivalent to optimal clustering structure (see Section 2). Our methods are general and are applicable to a wide class of graph models. For the preferential attachment model, a recent work [14] considers partial order inference, but the methods are specific to that model and not extendable.

Main notation. In the following π always represents a uniform random permutation from S_n, the symmetric group on n letters and n will be implied from the context. Let $G_n, \mathcal{G}_n, \mathbb{G}_n$ be the deterministic graph, random graph from the model under consideration and the set of graphs with n nodes. Similar definition holds for H_n and $\mathcal{H}_n$, but with $\mathcal{H}_n = \pi(\mathcal{G}_n)$. We label the vertices in the original graph G in their arrival order, $[n] = \{1, ..., n\}$, where node j is the jth node to arrive. For a Markov chain $\{X_k\}_{k\geq 0}$ with transition probability matrix P, $\mathbb{E}_x[\cdot]$ indicate expectation with respect to its sample paths starting from $X_0 = x$.

2 PROBLEM FORMULATION

Let H_n be the observed undirected and unweighted graph of n nodes with $V(H_n)$ as the set of vertices and $E(H_n)$ as the set of edges. The graph H_n is evolved over time, starting from a seed graph H_{n_0} with n_0 nodes. At a time instant k, when a new node appears and a set of new edges are added with one endpoint at the new node, the graph H_k will evolve into H_{k+1}. Since the change in graph structure occurs only when a new node is added, assuming the addition of a node as a time epoch, H_n also represents graph at time epoch n.

Given only one snapshot of the dynamic graph H_n, we usually do not know the time or order of arrivals of nodes. Our goal is to label each node with a number i, $1 \leq i \leq K$, such that all the nodes labeled by i arrived before nodes with labels j where $j > i$. The number of clusters (labels) K is unknown a priori and is a part of the optimal clustering formulation. Let C_i denote the set of nodes which carry label i. We note here that unlike the classical clustering, on top of the objective of the clustering, these clusters are ordered such that $C_1 \prec C_2 \ldots \prec C_K$, where the relation $C_i \prec C_j, i < j$ is defined as all the nodes inside the cluster C_i are estimated to be arrived earlier than all the nodes in the cluster C_j, and all the nodes inside each cluster are considered to be identical in arrival order.

The arrival of a new node and the strategy it uses to choose the existing nodes to make connections depend on the graph generation model. We thus express the above formulation in the following way. Let G_n be a graph drawn from a dynamic random graph model $\mathcal{G}_n$ on n vertices in which nodes are labeled $1, 2, \ldots, n$ according to their arrival, i.e., node j was the jth node to arrive. Let G_n be evolved from the seed graph G_{n_0}[1]. To model the lack of knowledge of the original labels, we subject the nodes to a permutation π drawn uniformly at random from the symmetric group on n letters

[1]The time epoch t_0 denotes the creation of G_{n_0} or H_{n_0} graph

S_n, and we are given the graph $H_n := \pi(G_n)$; that is, the nodes of G_n are randomly *relabeled*. Our original goal is to infer the arrival order in G_n after observing H_n, i.e., to find π^{-1}. The permutation π^{-1} gives the true arrival order of the nodes of the given graph.

Relation between temporal order clustering and partial order set. Instead of putting a constraint on recovering the whole permutation π^{-1} (or equivalently $K = n$ clusters), we resort to strict partial orders (without reflexive condition). For a partial order set σ, a relation $u <_\sigma v$ means that u is less than v in σ, that is, node u is older than node v. Every partial order σ can be represented by bins (clusters) $\{C_i\}$) as follows. A strict partial order set can be represented initially by a directed acyclic graph (DAG) with nodes as the nodes in the graph H_n and directed edges as given by σ: an edge from v to u exists when $u <_\sigma v$. Then taking the transitive closure of this DAG will result in the DAG of the partial order set σ. Now, all the nodes with in-degree 0 in the DAG will be part of cluster C_K and the set of nodes with all the in-edges coming from nodes in C_K will form cluster C_{K-1}. This process repeats until we get C_1. The number of clusters K is not defined before but found from the DAG structure.

We define an estimator ϕ of the temporal ordered clustering as a function $\phi : \mathbb{G}_n \to \mathbb{S}_n$, where $\mathbb{S}_n$ is the set of all partial orders of nodes $1, \ldots, n$. We consider estimators based on *unsupervised* and *semi-supervised* algorithms:

- Unsupervised: In this case, the node arrival order estimator does not assume any prior knowledge. Since the estimator should be designed incorporating intricacies of a graph model, its results will be based only on the assumption that the graph model fits well the real-world network under consideration.
- Semi-supervised: In some of the real-world networks, partial information of the order of nodes is known. Taking this information into account would help the estimator to more adapt to the real-data. We call such information as *perfect pairs*, that exists with probability 1. The estimator learns the partial orders in the data without violating the perfect pairs.

Measures for evaluating partial order. For a partial order σ, let $K(\sigma)$ denote the number of pairs (u, v) that are comparable under σ: i.e., $K(\sigma) = |\{(u, v) \ : \ u <_\sigma v\}|$, where $|K(\sigma)| \leq \binom{n}{2}$.
Density: the density of a partial order σ is simply the number of comparable pairs, normalized by the total possible number, $\binom{n}{2}$. That is, $\delta(\sigma) = \frac{K(\sigma)}{\binom{n}{2}}$. Note that $\delta(\sigma) \in [0, 1]$. Then the density of a partial order estimator ϕ is simply its minimum possible density $\delta(\phi) = \min_H[\delta(\phi(H))]$.
Measure of precision: it measures the expected fraction of *correct* pairs out of all pairs that are guessed by the partial order. That is

$$\theta(\sigma) = \mathbb{E}\left[\frac{1}{K(\sigma)} |\{u, v \in [n] \ : \ u <_\sigma v, \pi^{-1}(u) < \pi^{-1}(v)\}|\right].$$

For an estimator ϕ, we also denote by $\theta(\phi)$ the quantity $\mathbb{E}[\theta(\phi(\pi(G)))]$. We note here that the typical graph clustering performance measures like Silhouette index and Davies–Bouldin index do not find useful in our set up since the distance measure in our case is difficult to capture quantitatively and is purely based on indistinguishability due to symmetries and arrival order of nodes.

3 SOLVING THE OPTIMIZATION PROBLEM

The precision of a given estimator ϕ can be written in the form of a sum over all graphs H:

$$\theta(\phi) = \sum_H \Pr[\pi(\mathcal{G}) = H] \frac{1}{K(\phi(H))} \times \mathbb{E}\left[|\{u, v \in [n] : u <_{\phi(H)} v, \pi^{-1}(u) < \pi^{-1}(v)\}| \Big| \pi(\mathcal{G}) = H\right]$$

Here π and $\mathcal{G}$ are the random quantities in the conditional expectation. We formulate the optimal estimator as the one that gives maximum precision for a given minimum density. For an estimator to be optimal, it is then sufficient to choose, for each H, a partial order $\phi(H)$ that maximizes the expression

$$J_\varepsilon(\phi) := K(\phi(H))^{-1} \times \mathbb{E}\left[|\{u, v \in [n] \ : \ u <_{\phi(H)} v, \pi^{-1}(u) < \pi^{-1}(v)\}| \Big| \pi(\mathcal{G}) = H\right]$$

subject to the density constraint that $K(\phi(H)) \geq \varepsilon\binom{n}{2}$.

3.1 Integer programming formulation

In this subsection we recall and extend some results from a recent work in [14]. We now represent the optimization problem with $J_\varepsilon(\phi)$ as an integer program (IP). For an estimator ϕ, we define a binary variable $x_{u,v}$ for each ordered pair (u, v) as $x_{u,v} = 1$ when $u <_{\phi(H)} v$. Note that $x_{u,v} = 0$ means either $u >_{\phi(H)} v$ or the pair (u, v) is incomparable in the partial order $\phi(H)$. Let $p_{u,v}(H) = \Pr[\pi^{-1}(u) < \pi^{-1}(v) | \pi(\mathcal{G}) = H]$ is the probability that u arrived before v given the relabeled graph H.

In the following, we write the optimization in two forms: the original integer program (left) and the linear programming approximation (right). the objective functions of both the formulations are equivalent to $J_\varepsilon(\phi)$. The constraints of the optimizations correspond to domain restriction, minimum density, and partial order constraints - antisymmetry and transitivity respectively. To use a linear programming (LP) approximation, we first convert the rational integer program into an equivalent truly integer program. With the substitution $s = 1/\sum_{1 \leq u \neq v \leq n} x_{u,v}$, and $y_{u,v} = s x_{u,v}$), the objective function rewritten as a linear function of the normalized variables. The new variables $y_{u,v}$ take values from $\{0, s\}$. In fact, taking $s = 1/\varepsilon\binom{n}{2}$ is enough for the equivalence. For the LP relaxation, we assume the domain of $y_{u,v}$ as $\left[0, 1/\varepsilon\binom{n}{2}\right]$.

Original integer program	**LP approximation**
$\max_x \frac{\sum_{1 \leq u \neq v \leq n} p_{u,v}(H) x_{u,v}}{\sum_{1 \leq u \neq v \leq n} x_{u,v}}$	$\max_y \sum_{1 \leq u \neq v \leq n} p_{u,v}(H) y_{u,v}$
subject to	subject to (let $s := 1/\varepsilon\binom{n}{2}$)
$x_{u,v} \in \{0, 1\}, \forall u, v \in [n]$	$y_{u,v} \in [0, s], \forall u, v \in [n]$
$\sum_{1 \leq u \neq v \leq n} x_{u,v} \geq \varepsilon\binom{n}{2}$	$\sum_{1 \leq u \neq v \leq n} y_{u,v} = 1$
$x_{u,v} + x_{v,u} \leq 1, \forall u, v \in [n]$	$y_{u,v} + y_{v,u} \leq s, \forall u, v \in [n]$
$x_{u,w} \geq x_{u,v} + x_{v,w} - 1,$ $\forall u, v, w \in [n].$	$y_{u,v} + y_{v,w} - y_{u,w} \leq s,$ $\forall u, v, w \in [n].$

The next lemma bounds the effect of approximating the coefficients p_{uv} on the optimal value of the integer program.

Lemma 1. *Consider the integer program whose objective function is given by* $\hat{J}_{\varepsilon,\lambda}(\phi) = \frac{\sum_{1 \leq u < v \leq n} \hat{p}_{u,v}(H) x_{u,v}}{\sum_{1 \leq u \neq v \leq n} x_{u,v}}$, *with the same constraints*

as in the original IP. Assume $p_{u,v}(H)$ can be approximated with $|\hat{p}_{u,v}(H) - p_{u,v}(H)| \leq \lambda$ uniformly for all u, v. Let ϕ_* and $\hat{\phi}_*$ denote optimal points for the original and modified integer programs, respectively. Then $|\hat{J}_{\varepsilon,\lambda}(\hat{\phi}_*) - J_\varepsilon(\phi_*)| \leq 3\lambda$, for arbitrary $\lambda > 0$.

The proof of the above lemma is an extension of a similar lemma in [14]. But in [14], the authors assume $|\hat{p}_{uv}/p_{uv} - 1| \leq \lambda$, which is a stronger condition than our present assumption $|\hat{p}_{uv} - p_{uv}| \leq \lambda$.

3.2 Estimating coefficients of optimal precision integer program

Let $\Gamma(H_t)$ be the set of all permutations σ which has a positive probability for $\sigma(H_t)$ under the graph generation model. We have, for $p_{uv} := \mathbb{P}(\pi^{-1}(u) < \pi^{-1}(v) | \pi(\mathcal{G}_t) = H_t)$,

$$\begin{aligned} p_{uv} &= \sum_{\substack{\sigma\,:\,\sigma^{-1}\in\Gamma(H) \\ \sigma^{-1}(u)<\sigma^{-1}(v)}} \mathbb{P}(\pi = \sigma | \pi(\mathcal{G}_t) = H_t) \\ &= \sum_{\substack{\sigma\,:\,\sigma^{-1}\in\Gamma(H_t) \\ \sigma^{-1}(u)<\sigma^{-1}(v)}} \frac{\mathbb{P}[\pi = \sigma, \pi(\mathcal{G}_t) = H_t]}{\mathbb{P}[\pi(\mathcal{G}_t) = H_t]} \\ &= \frac{\sum_{\sigma\,:\,\sigma^{-1}\in\Gamma(H_t)} \mathbb{P}[\mathcal{G}_t = \sigma^{-1}(H_t)]}{\sum_{\sigma^{-1}\in\Gamma(H_t)} \mathbb{P}[\mathcal{G}_t = \sigma^{-1}(H_t)]}, \end{aligned} \tag{1}$$

where we used $\mathbb{P}[\pi = \sigma]$ independent of H_t as $1/n!$.

A Markov chain approach to approximate p_{uv}. Let $\mathcal{R}_{H_t} \subset V(H_t)$ be the set of *possible candidates* for youngest nodes at time t, and let $\mathcal{P}_{H_t}(v)$, represents possible parents of v in H_t, i.e., the nodes v selects for duplication. The set $\mathcal{R}_{H_t} \subset V(H_t)$ depends on the graph model. For e.g., in case of preferential attachment model, in which a new node attaches m edges to the existing nodes with a probability distribution proportional to the degree of the existing node, $\mathcal{R}_{H_t}$ is the set of m-degree nodes. We consider only permutations that do not change the initial graph G_{t_0} labels. For example, if G_{t_0} has three nodes and G_t has 6 nodes, we consider the following permutations (represented in cyclic notation): (1)(2)(3)(456), (1)(2)(3)(45)(6), (1)(2)(3)(46)(5), (1)(2)(3)(4)(56), (1)(2)(3)(4)(5)(6). Thus we define H_{t_0} as G_{t_0} itself.

Let $\delta(H_t, v_t)$ represent the graph in which node v_t is deleted from H_t, where $v_t \in \mathcal{R}_{H_t}$. Then the graph sequence $\mathcal{H}_t = H_t, \mathcal{H}_{t-1} = \delta(H_t, v_t), \ldots, \mathcal{H}_{t_0} = H_{t_0}$ forms a nonhomogeneous Markov chain – nonhomogeneous because the state space $\{\mathbb{H}_s\}_{s\leq t}$ changes with s and thus the transition probabilities too. Similarly $\mathcal{G}_t, \mathcal{G}_{t-1}, \mathcal{G}_{t_0}$ also make a Markov chain, and for a fixed permutation σ, $\sigma(G) = H$, both the above Markov chains have same transition probabilities.

Note that the posterior probability of producing H_t from $\delta(H_t, v_t)$ is given by

$$w(\delta(H_t, v_t), H_t) := \mathbb{P}[\mathcal{H}_t = H_t | \mathcal{H}_{t-1} = \delta(H_t, v_t)]$$

The following theorem characterizes our estimator.

Theorem 1. *Consider a time-nonhomogeneous Markov chain $\mathcal{H}_t = H_t, \mathcal{H}_{t-1} = \delta(H_t, \mathcal{U}_t), \ldots$, where $\mathcal{U}_t \in \mathcal{R}_{H_t}, \mathcal{U}_{t-1} \in \mathcal{R}_{H_{t-1}}, \ldots$ be the nodes removed randomly by the Markov chain and let its transition probability matrices be $\{Q_s = [q_s(\tilde{H}_i, \tilde{H}_j)]\}_{s\leq t}$ with $\tilde{H}_i \in \mathbb{G}_s$ and $\tilde{H}_j \in \mathbb{G}_{s-1}$. Then we have*

$$\sum_{\sigma^{-1}\in\Gamma(H_t)} \mathbb{P}[\mathcal{G}_t = \sigma^{-1}(H_t) | H_0] = \mathbb{E}_{\mathcal{H}_t = H_t}\left[\prod_{s\leq t}^{t_0+1} \frac{w(\delta(H_s, \mathcal{U}_s), H_s)}{q_s(H_s, \delta(H_s, \mathcal{U}_s))}\right].$$

Proof. Now we have the following iterative expression for the denominator of p_{uv}.

$$\begin{aligned} p_{uv}^{\text{denom}}(H_t, H_{t_0}) &:= \sum_{\sigma^{-1}\in\Gamma(H_t)} \mathbb{P}[\mathcal{G}_t = \sigma^{-1}(H_t) | H_{t_0}] \qquad (2) \\ &= \sum_{v_t\in\mathcal{R}_{H_t}} \sum_{\sigma^{-1}\in\Gamma(H_t)} \mathbb{P}[\mathcal{G}_t = \sigma^{-1}(H_t), \mathcal{G}_{t-1} = \sigma_1^{-1}(\delta(H_t, v_t)) | H_{t_0}], \end{aligned}$$

where $\sigma_1 \in S_{n-1}$ is the permutation σ with "v_t maps to n" removed. Now we can rewrite the above expression as

$$\begin{aligned} \sum_{v_t\in\mathcal{R}_{H_t}} \sum_{\sigma^{-1}\in\Gamma(H_t)} &\mathbb{P}[\mathcal{G}_t = \sigma^{-1}(H_t) | \mathcal{G}_{t-1} = \sigma_1^{-1}(\delta(H_t, v_t))] \\ &\times \mathbb{P}[\mathcal{G}_{t-1} = \sigma_1^{-1}(\delta(H_t, v_t)) | H_{t_0}]. \end{aligned}$$

Note that $\mathbb{P}[\mathcal{G}_t = \sigma^{-1}(H_t) | \mathcal{G}_{t-1} = \sigma_1^{-1}(\delta(H_t, v_t))]$ for a fixed σ (thus σ_1) is equivalent to $w(\delta(H_t, v_t), H_t)$. Now introducing a transition probability $\{Q_s = [q_s(i,j)]\}_{s\leq t}$ for the Markov chain $\{H_s\}_{s\leq t}$, and using importance sampling,

$$\begin{aligned} p_{uv}^{\text{denom}}(H_t, H_{t_0}) &= \sum_{v_t\in\mathcal{R}_{H_t}} \frac{w(\delta(H_t, v_t), H_t)}{q_t(H_t, \delta(H_t, v_t))} q_t(H_t, \delta(H_t, v_t)) \\ &\quad \times \sum_{\sigma^{-1}\in\Gamma(\delta(H_t, v_t))} \mathbb{P}[\mathcal{G}_{t-1} = \sigma^{-1}(\delta(H_t, v_t)) | H_{t_0}]. \\ &= \sum_{v_t\in\mathcal{R}_{H_t}} \frac{w(\delta(H_t, v_t), H_t)}{q_t(H_t, \delta(H_t, v_t))} q_t(H_t, \delta(H_t, v_t)) \\ &\quad \times p_{uv}^{\text{denom}}(\delta(H_t, v_t), H_{t_0}), \end{aligned} \tag{3}$$

with $p_{uv}^{\text{denom}}(H_{t_0}, H_{t_0}) = 1$. Here $q_t(H_t, \delta(H_t, v_t))$ is the transition probability to jump from $\mathcal{H}_t = H_t$ to $\mathcal{H}_{t-1} = \delta(H_t, v_t)$.

$$\text{Now let } \mu(H_t, H_{t_0}) = \mathbb{E}_{\mathcal{H}_t = H_t}\left[\prod_{s\leq t}^{t_0+1} \frac{w(\delta(H_s, \mathcal{U}_s), H_s)}{q_s(H_s, \delta(H_s, \mathcal{U}_s))}\right]$$

Then we have,

$$\begin{aligned} \mu(H_t, H_{t_0}) &= \sum_{v_t\in\mathcal{R}_{H_t}} \frac{w(\delta(H_t, v_t), H_t)}{q_t(H_t, \delta(H_t, v_t))} q_t(H_t, \delta(H_t, v_t)) \\ &\quad \times \mathbb{E}_{\mathcal{H}_{t-1}=\delta(H_t, v_t)}\left[\prod_{s\leq t-1}^{t_0+1} \frac{w(\delta(H_s, \mathcal{U}_s), H_s)}{q_s(H_s, \delta(H_s, \mathcal{U}_s))}\right] \qquad (4) \\ &= \sum_{v_t\in\mathcal{R}_{H_t}} \frac{w(\delta(H_t, v_t), H_t)}{q_t(H_t, \delta(H_t, v_t))} q_t(H_t, \delta(H_t, v_t)) \\ &\quad \times \mu(\delta(H_t, v_t), H_{t_0}), \qquad (5) \end{aligned}$$

where (4) follows from the Markov property.

Defining the function at t_0 as

$$\frac{w(\delta(H_{t_0}, v_{t_0}), H_{t_0+1})}{q_{t_0}(H_{t_0+1}, \delta(H_{t_0}, v_{t_0}))} = 1, \text{ for any } v_{t_0}$$

we note here that the iteration (5) of $\mu(H_t, H_{t_0})$ is identical to that of p_{uv}^{denom} in (3). This completes the proof. □

Note that unlike $q_t(H_t, \delta(H_t, v_t))$, which is under our control to design a Markov chain, $w(\delta(H_t, v_t), H_t)$ is a well-defined fixed quantity (see (7)). The only constraint for the transition probability matrices $\{Q_s\}_{s \le t}$ is that it should be chosen to be in agreement with the graph evolution such that the choices of jumps from H_s to H_{s-1} restricts to removing nodes from $\mathcal{R}_{H_s}$. In other words, the Theorem 1 gives a sequential importance sampling strategy with normalized terms.

Now we can form the estimator for p_{st} for a node pair (s, t) as follows. Let $\boldsymbol{v}^{(k)}$ be the vector denoting the sampled node sequence of the kth run of the Markov chain. It can either represent a vector notation as $\boldsymbol{v}^{(k)} = (v_t^{(k)}, v_{t-1}^{(k)}, \ldots, v_{t_0+1}^{(k)})$ or take a function form $\boldsymbol{v}^{(k)}(s)$ denoting the new label of a vertex s in H. The estimator $\hat{p}_{st}^{(k)}$ is now, for all $s, t \in H$

$$\hat{p}_{st}^{(k)} = \frac{\sum_{i=1}^{k} \mathbf{1}_{\{\boldsymbol{v}^{(i)}(s) < \boldsymbol{v}^{(i)}(t)\}} \prod_{s \le t}^{t_0+1} \frac{w(\delta(H_s, \boldsymbol{v}_s^{(i)}), H_s)}{q_s(H_s, \delta(H_s, \boldsymbol{v}_s^{(i)}))}}{\sum_{i=1}^{k} \prod_{s \le t}^{t_0+1} \frac{w(\delta(H_s, \boldsymbol{v}_s^{(i)}), H_s)}{q_s(H_s, \delta(H_s, \boldsymbol{v}_s^{(i)}))}}. \tag{6}$$

Using continuous mapping theorem, we can prove that $\hat{p}_{st}^{(k)} \to p_{st}$ a.s. as $k \to \infty$.

4 APPROXIMATING OPTIMAL SOLUTION

In this section, we describe our main algorithms for node arrival order recovery of a general graph model.

Algorithms for sampling the Markov chain. Finding the whole set of permutations and calculating the exact p_{uv} according to (1) is of exponential complexity. With Theorem 1 and eq. (6), we can approximate p_{uv} as the empirical average of Markov chain based sample paths. We try two different importance sampling distributions $\{Q_s\}_{s \le t}$:

- `local-unif-sampling` with transition probabilities

$$q_s(H_s, \delta(H_s, v_s)) = \frac{1}{|\mathcal{R}_{H_s}|}.$$

- `high-prob-sampling` forms the Markov chain with

$$q_s(H_s, \delta(H_s, v_s)) = \frac{w(\delta(H_s, v_s), H_s)}{\sum_{u \in \mathcal{R}_{H_s}} w(\delta(H_s, u), H_s)}.$$

  The above transition probability corresponds to choosing the high probability paths.

Though the `high-prob-sampling` looks like the right approach to follow, as we show later in Section 6.1, it has much slower rate of convergence than `local-unif-sampling`. Moreover at each step s, `high-prob-sampling` requires $O(n^2)$ computations, while `local-unif-sampling` requires only $O(n)$.

The `local-unif-sampling` can be further improved with the acceptance-rejection sampling technique: at a step t, randomly sample a node u from $V(H_t)$ (instead of sampling from $\mathcal{R}_{H_t}$). Then calculate the probability that the node u be the youngest node in the graph. If this probability is positive, we accept u as V_t and if it is zero, we randomly sample again from $V(H_t)$.

Now we assume that p_{uv} are estimated for all u and v to propose algorithms for temporal clustering. In fact, according to Lemma 1, we only need to have $\max_{u,v} |\hat{p}_{uv} - p_{uv}| \le \lambda$ for a small $\lambda > 0$. Thus for small p_{uv}, $\hat{p}_{uv}$ can be assumed to be zero. We propose the following unsupervised and semi-supervised algorithms based on the estimates of p_{uv}.

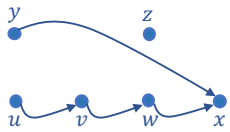


Figure 2: Supervised learning example DAG for $\sigma_{train} = \{(u, v), (v, w), (w, x), (y, w)$: $\mathcal{NR}_{H_t} = \{v, w, x\}$ and $\mathcal{R}_{H_t} = \{u, y, z\}$.

4.1 Unsupervised solution

`sort-by-`p_{uv}`-sum` algorithm. For this algorithm, we construct a new complete graph with the node set same as that of H_n and edge weights as p_{uv}. Let us now define a metric $p_u := \sum_{v \in V} p_{uv}$ for every node u of H_n. Since p_{uv} denotes the probability that node u is older than node v, p_u would give a high score when a node u becomes the oldest node. Our ranking is then sorted order of the p_u values.

Instead of total order, a partial order can be found by a simple binning over p_u values: fix the bin size $|C|$ and group $|C|$ nodes in the sorted p_u values into a cluster, and the process repeats for other clusters. If $|C| = 1$, the algorithm will yield a total order.

p_{uv}`-threshold` algorithm. Here, each of the estimated p_{uv}'s is compared against a threshold τ. Only the node pairs that are strictly greater than this condition are put into the estimator output partial order. Note that if $\tau = 0.5$, we get a total order in virtually all relevant cases.

4.2 Semi-supervised solution

Suppose we have partial true data available. Let it be ordered in partial order as $\sigma_{\text{orig}} = \{(u, v)\}$, in which for the pair (u, v), u is the older than v. Let $\sigma_{\text{train}} \subset \sigma_{\text{orig}}$ be the training set and the let the test set be $\sigma_{\text{test}} := \sigma_{\text{orig}} \backslash \sigma_{\text{train}}$. Let $|\sigma_{\text{train}}| = \alpha |\sigma_{\text{orig}}|$ for some $0 < \alpha < 1$. With the knowledge of σ_{train}, we modify the the estimation of p_{uv} as follows. The set of removable nodes $\mathcal{R}_{H_t}$ at each instant is modified to $\mathcal{R}_{H_t} \backslash \mathcal{NR}_{H_t}$, where $\mathcal{NR}_{H_t}$ is the set of nodes that can not be included in the removable nodes as it would violate the partial order of σ_{train}. It is defined as follows:

$$\mathcal{NR}_{H_t} := \{u : (u, v) \in \sigma_{\text{train}}, u, v \in V(H_t)\}.$$

After estimating p_{uv} with the redefined $\mathcal{R}_{H_t}$, we employ `sort-by-`p_{uv}`-sum` algorithm and p_{uv}`-threshold` algorithms to find partial order. An example of $\mathcal{R}_{H_t}$ construction is shown in Figure 2.

5 TEMPORAL ORDER CLUSTERING FOR DUPLICATION-DIVERGENCE MODEL

5.1 Duplication-divergence model (DD-model)

We consider Pastor-Satorras et al. definition of the DD-model [12]. It proceeds as follows. Given an undirected, simple seed graph G_{n_0} on n_0 nodes and target number of nodes n, the graph G_{k+1} with $k+1$ nodes[2] evolves from the G_k as follows: first, a new vertex v is added to G_k. Then the following steps are carried out:

- Duplication: Select an node u from G_k uniformly at random. The node v then makes connections to $\mathcal{N}(u)$, the neighbor set of u.

[2]The subscript k with G_k can also be interpreted as time instant k

- Divergence: Each of the newly made connections from v to $\mathcal{N}(u)$ are deleted with probability $1-p$. Furthermore, for all the nodes in G_k to which v is not connected, create an edge from it to v independently with probability $\frac{r}{k}$.

The above process is repeated until the number of nodes in the graph is equal to n. We denote the graph G_n generated from the DD-model with parameters p and r, starting from seed graph G_{n_0}, by $G_n \sim \texttt{DD-model}(n, p, r, G_{n_0})$.

For a node $v_t \in V(H_t)$, the probability of having the node u as its parent in the above model is defined as

$$w(\delta(H_t, v_t), u, H_t) = \frac{1}{t-1} p^{|\mathcal{N}(v_t)\cap\mathcal{N}(u)|}(1-p)^{|\mathcal{N}(u)\setminus\mathcal{N}(v_t)|} \left(\frac{r}{t-1}\right)^{|\mathcal{N}(v_t)\setminus\mathcal{N}(u)|} \left(1-\frac{r}{t-1}\right)^{(t-1)-|\mathcal{N}(v_t)\cup\mathcal{N}(u)|} \quad (7)$$

Then $w(\delta(H_t, v_t), H_t) = \sum_{u\in\mathcal{P}_{H_t}(v_t)} w(\delta(H_t, v_t), u, H_t)$.

Since all permutations have positive probability in this version of the model, we have $\mathcal{R}_{H_t} = V(H_t)$ and $\Gamma(H_t) = t!$.

5.2 Greedy algorithms

To form a comparison with algorithms proposed in Section 4, we propose the following greedy algorithms for the DD-model.

`sort-by-degree`. The nodes are sorted by the degree and arranged into clusters. Cluster C_1 contains nodes with the largest degree.

`peel-by-degree`. The nodes with the lowest degree are first collected and put in the highest cluster. Then they are removed from the graph, and the nodes with the lowest degree in the remaining graph are found and the process repeats.

`sort-by-neighborhood`. This algorithm will output a partial order with all ordered pairs $(u \prec v)$ such that $\mathcal{N}(u)$ contains $\mathcal{N}(v)$. This condition holds when $r = 0$. When $r > 0$, we consider $|\mathcal{N}(v)\setminus N(u)| \leq r$ as r is the average number of *extra* connections a node makes apart from duplication process. In most real-world data, we estimate r as smaller than 1, and hence the original check is sufficient.

`peel-by-neighborhood`. Here, we find the set $\{u : \nexists v |\mathcal{N}(u)\setminus\mathcal{N}(v)| \leq r\}$ (as mentioned before, it is sufficient to check $\mathcal{N}(u) \subset \mathcal{N}(v)$ in many practical cases) and mark it as the youngest cluster. These nodes are removed from the graph, and the process is repeated until it hits G_0. This algorithm makes use of the DAG of the neighborhood relationship and includes isolated nodes into the bins.

5.3 Comparison with other graph models

The node arrival order recovery problem in the DD-model is different from that in other graph models like Erdős-Renyi graphs and preferential attachment graphs.

First, for a fixed graph H_n on n vertices, let us consider a set of graphs $\Gamma(H) = \{\sigma(H) : \sigma \in S_n\}$. It is obvious that for the Erdős-Renyi model, any graph in $\Gamma(H)$ is generated equally likely with a given seed graph G_{n_0}. Such property was also proved for the preferential attachment model in [10]. However, this does not hold for DD-model graphs as shown in the following example.

For the graph G_n presented in Fig. 3, let G_{n_0} consists of vertices 1, 2, 3, and let the parameters of the DD-model be $p = 0.2$ and $r = 0$. The total probability of generating a structure (ignoring labels) identical to G_t is equal to 0.224. Therefore, from eq. 7, the probability of adding vertices in order (1, 2, 3, 4, 5) is 0.7714, but the probability of adding vertices in order (1, 2, 3, 5, 4) is only 0.2285.

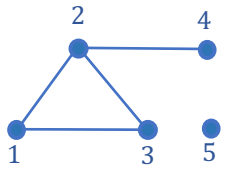


Figure 3: Example asymmetric graph

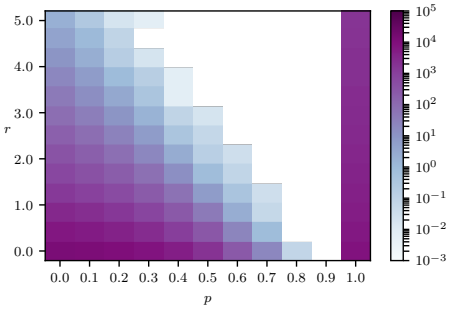


Figure 4: $\mathbb{E}\log|\mathrm{Aut}(G_t)|$, $G_t \sim$ `DD-model`$(2000, p, r, K_{20})$, where $|\mathrm{Aut}(G_t)|$ is the number of automorphisms in graph G_t.

Second, it is well known that both the Erdős-Renyi graphs and preferential attachment graphs are asymmetric with high probability [5, 10]. On the other hand, the graphs generated from the DD-model for a certain range of parameters show a significant amount of symmetry, as shown in Fig. 4. This is in accordance with many real-world networks (see Table 3 for examples).

Last, the behavior of the average degree at time t of the node arrived at an earlier time s (denoted by $\deg_s(t)$) is different for all three models. For Erdős-Renyi graph with edge probability p, it is known that $\mathbb{E}[\deg_s(t)] = p(t-1)$. For the preferential attachment graphs, $\mathbb{E}[\deg_s(t)] = \Theta\left(\sqrt{t/s}\right)$ ([18], Theorem 8.2). However, for the DD-model, $\mathbb{E}[\deg_s(t)] = \Theta\left((t/s)^p s^{2p-1}\right)$ for any $t \geq s$ [17]. Note that when $s = O(1)$ - the case of very old nodes - the degree is of order t^p. And for $s = t$, we have $\mathbb{E}[\deg_t(t)] = O(t^{2p-1})$ which only for $p > 1/2$ is growing. For example when $p = 1$ all degrees on average are of order $O(t)$. Thus oldest nodes in the graph need not have large average degrees as the graph evolves, and algorithms based on such a heuristic are not applicable for the DD-model.

6 EXPERIMENTS

In this section, we evaluate our methods on synthetic and real-world data. We made publicly available all the code and data of this project at https://bit.ly/308XN9m. The numerical comparisons with the real-world data in this section are with respect to the greedy heuristic algorithms since there are no other existing techniques to estimate temporal ordered clustering, according to our knowledge. For deriving total order, a natural solution will be the maximum likelihood estimator (MLE). But we do not consider MLE explicitly here because it is known that many networks exhibit large number of symmetries (see Table 3 for example), and thus there will be large number of total orders that achieves the MLE criterion with low value of precision. In fact, our optimal formulation in Section 3

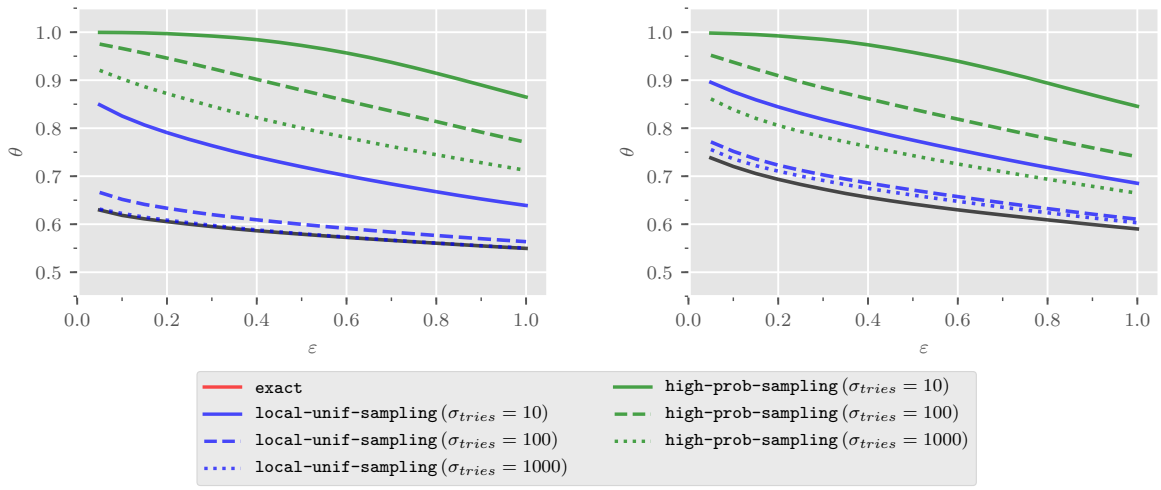


Figure 5: Results on synthetic networks with exact curve: $G_n \sim$ `DD-model`$(13, p, 1.0, G_{n_0})$ for $p = 0.3$ (left) and 0.6 (right), averaged over 100 graphs. G_{n_0} is generated from Erdős-Renyi graph with $n_0 = 4$ and $p_0 = 0.6$.

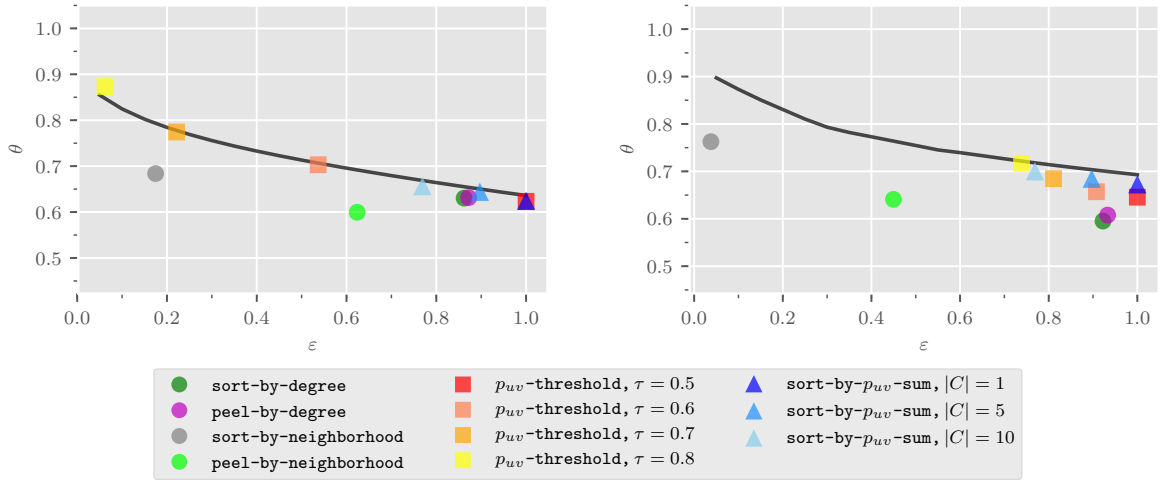


Figure 6: Results on synthetic networks with greedy and *unsupervised learning* p_{uv}-based algorithms: $G_n \sim$ `DD-model`$(50, p, 1.0, G_{n_0})$ for $p = 0.3$ (left) and 0.6 (right), averaged over 100 graphs. p_{uv}-based algorithms use $\sigma_{\text{tries}} = 100{,}000$. G_{n_0} is generated from Erdős-Renyi model with $n_0 = 10$ and $p_0 = 0.6$. The theoretical curve is estimated via `local-unif-sampling`.

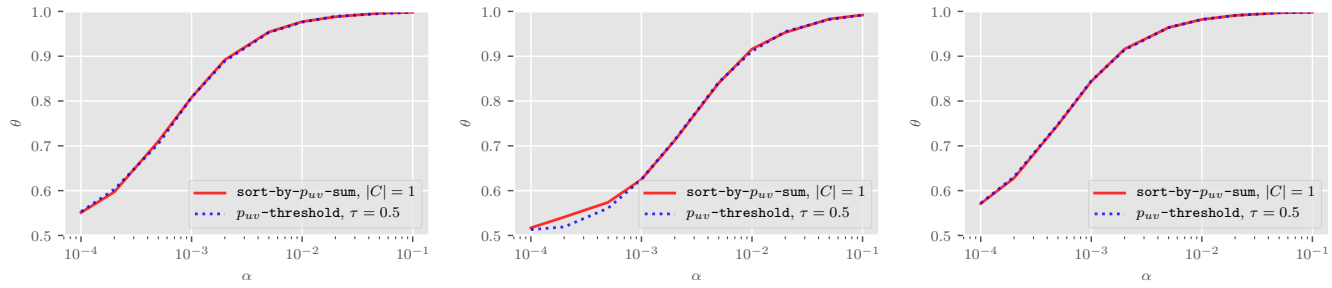
already captures the MLE solutions and outputs it if it achieves high precision.

6.1 Synthetic data

In Figures 5 and 6, we provide the linear programming (LP) *optimal curve* or an approximation to it by sampling. The LP optimal curve is given by the optimal precision values computed from the relaxed LP formulation and exact p_{uv} values for various values of ε, which corresponds to the density lower bound of $\varepsilon\binom{n}{2}$ (see Section 3.1 for more details). In all the figures σ_{tries} denote the number of Markov chain sample paths considered for sampling.

Figure 5 shows the convergence of the approximated optimal curve obtained through different sampling methods to the exact LP optimal curve. We observe that the convergence of the estimated curve is highly dependent on the method of estimation: `local-unif-sampling` method requires only 100 samples, but `high-prob-sampling` is still visibly far away from LP optimal curve even for 1000 samples. We consider a small size example here since it becomes infeasible ($\Theta(n!)$) to compute the exact curve for larger values. Thus, along with the computational reasons stated in Section 4, we use `local-unif-sampling` in the subsequent experiments.

In Figure 6, we present the results of the unsupervised algorithms and its comparison with the estimated optimal curve via `local-unif-sampling`. It turns out that greedy algorithms perform reasonably well for small p, but their performance deteriorates for higher values of p. On the other hand, p_{uv}-based algorithms (`sort-by-`p_{uv}`-sum` and p_{uv}`-threshold`) offer consistent, close to the theoretical bound, behavior for the whole range of p. Moreover, both bin size in `sort-by-`p_{uv}`-sum` and threshold in



(a) ArXiv network (b) Simple English Wikipedia network (c) CollegeMsg network

Figure 7: Real-world networks: results of semi-supervised learning

p_{uv}-threshold algorithm offer a trade-off between higher precision and higher density. It is worth noting that the free parameters in both p_{uv}-based algorithms serve as a trade-off: the higher the threshold or the greater the bin size, we observe a decrease in density, but increase in precision as we stay close to the theoretical curve.

Table 1 contains the results of semi-supervised learning for p_{uv}-based algorithms. Even a small increase in the percentage of the training set α produces a large precision increase for all sets of parameters. All the conclusions are summed up in Table 2.

		p = 0.3		p = 0.6	
Algorithm	α	δ	θ	δ	θ
sort-by-p_{uv}-sum, $\lvert C\rvert$= 1	0.001	1.0	0.598	1.0	0.613
sort-by-p_{uv}-sum, $\lvert C\rvert$= 1	0.01	1.0	0.643	1.0	0.650
sort-by-p_{uv}-sum, $\lvert C\rvert$= 1	0.1	1.0	0.836	1.0	0.832
sort-by-p_{uv}-sum, $\lvert C\rvert$= 10	0.001	0.769	0.605	0.769	0.626
sort-by-p_{uv}-sum, $\lvert C\rvert$= 10	0.01	0.768	0.661	0.767	0.660
sort-by-p_{uv}-sum, $\lvert C\rvert$= 10	0.1	0.758	0.864	0.759	0.859
p_{uv}-threshold, τ = 0.5	0.001	1.0	0.604	1.0	0.617
p_{uv}-threshold, τ = 0.5	0.01	1.0	0.637	1.0	0.649
p_{uv}-threshold, τ = 0.5	0.1	1.0	0.829	1.0	0.823
p_{uv}-threshold, τ = 0.9	0.001	0.010	0.906	0.028	0.871
p_{uv}-threshold, τ = 0.9	0.01	0.020	0.951	0.090	0.907
p_{uv}-threshold, τ = 0.9	0.1	0.521	0.966	0.559	0.960

Table 1: Results on synthetic networks with *semi-supervised learning* p_{uv}-based algorithms: $G_n \sim$ DD-model$(50, p, 1.0, G_{n_0})$, averaged over 100 graphs. p_{uv}-based algorithms use $\sigma_{\text{tries}} = 100,000$. G_{n_0} is Erdős-Renyi graph with $n_0 = 10$ and $p_0 = 0.6$.

Algorithm	Fixed	Free	Free	Free
sort-by-p_{uv}-sum	α	$\lvert C\rvert \nearrow$	$\delta \searrow$	$\theta \approx$
sort-by-p_{uv}-sum	$\lvert C\rvert$	$\alpha \nearrow$	$\delta \approx$	$\theta \nearrow$
p_{uv}-threshold	α	$\tau \nearrow$	$\delta \searrow$	$\theta \nearrow$
p_{uv}-threshold	τ	$\alpha \nearrow$	$\delta \nearrow$	$\theta \nearrow$

Table 2: Conclusions from synthetic data: how the metrics behave by fixing one of the parameters and keeping other free. The symbol ≈ indicates the changes are not significant.

6.2 Real-world networks

We consider three real-world networks with ground truth (temporal information) available. The directed networks are treated as undirected for our purposes.

The ArXiv network: It is a directed network with 7,464 nodes and 116,268 edges. Here the nodes are the publications in arXiv online repository of theoretical high energy physics, and the edges are formed when a publication cite another.

The Simple English Wikipedia dynamic network: A directed network with 10,000 nodes and 169,894 edges. Nodes represent articles and an edge indicates that a hyperlink was added.

CollegeMsg network: In this dataset of private message sent on an online social platform at University of California, Irvine, nodes represent users and an edge from u to v indicates user u sent a private message to user v at time t. Number of nodes is 1,899 and number of edges is 59,835.

Network (G_{obs})	$\log\lvert\text{Aut}(G_{\text{obs}})\rvert$	$\widehat{p}$	$\widehat{r}$
ArXiv	12.59	0.72	1.0
Wikipedia	1018.94	0.66	0.5
CollegeMsg	231.54	0.65	0.45

Table 3: Parameters of the real-world networks estimated using recurrence-based method from [15].

Figures 7a-7c show the result of semi-supervised learning. We note here that a small increase in the percentage of the training set (α) leads to a huge change in the precision. This also happens in synthetic data, and is caused by the large structural dependency within networks, unlike in classical machine learning data where often the data is assumed to be independent, and benefits us to get a near-perfect clustering (precision close to 1) given only 1% of labeled data. The semi-supervised learning helps us to obtain a significant improvement over greedy algorithms which resulted in precision of 0.47 to 0.63, for a non-negligible value of density – but that is not much better than random guessing (see Table 4).

	ArXiv		Wikipedia		CollegeMsg	
Greedy algorithm	δ	θ	δ	θ	δ	θ
sort-by-degree	0.98	0.47	0.96	0.59	0.92	0.63
peel-by-degree	0.98	0.46	0.96	0.59	0.92	0.63
sort-by-neighborhood	0.0001	0.51	0.03	0.60	0.01	0.78
peel-by-neighborhood	0.13	0.50	0.80	0.593	0.75	0.61

Table 4: Real-world networks: results for greedy algorithms.

REFERENCES

[1] Eric Bair. 2013. Semi-supervised clustering methods. *Wiley Interdisciplinary Reviews: Computational Statistics* 5, 5 (2013), 349–361.

[2] Sugato Basu, Arindam Banerjee, and Raymond Mooney. 2002. Semi-supervised clustering by seeding. In *International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 27–34.

[3] Robert Görke, Pascal Maillard, Christian Staudt, and Dorothea Wagner. 2010. Modularity-driven clustering of dynamic graphs. In *International Symposium on Experimental Algorithms*. Springer, Berlin, Heidelberg, 436–448.

[4] Derek Greene, Donal Doyle, and Padraig Cunningham. 2010. Tracking the evolution of communities in dynamic social networks. In *2010 International Conference on Advances in Social Networks Analysis and Mining*. IEEE, Washington, DC, USA, 176–183.

[5] Jeong Han Kim, Benny Sudakov, and Van Vu. 2002. On the asymmetry of random regular graphs and random graphs. *Random Structures & Algorithms* 21, 3-4 (2002), 216–224.

[6] Brian Kulis, Sugato Basu, Inderjit Dhillon, and Raymond Mooney. 2009. Semi-supervised graph clustering: a kernel approach. *Machine Learning* 74, 1 (2009), 1–22.

[7] Si Li, Kwok Pui Choi, Taoyang Wu, and Louxin Zhang. 2013. Maximum likelihood inference of the evolutionary history of a PPI network from the duplication history of its proteins. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 10, 6 (2013), 1412–1421.

[8] Fuchen Liu, David Choi, Lu Xie, and Kathryn Roeder. 2018. Global spectral clustering in dynamic networks. *Proceedings of the National Academy of Sciences* 115, 5 (2018), 927–932.

[9] Andreas Loukas and Pierre Vandergheynst. 2018. Spectrally Approximating Large Graphs with Smaller Graphs. In *International Conference on Machine Learning*. PLMR, Stockholm, Sweden, 3243–3252.

[10] Tomasz Łuczak, Abram Magner, and Wojciech Szpankowski. 2019. Asymmetry and structural information in preferential attachment graphs. *Random Structures and Algorithms* (2019), 1–23.

[11] Saket Navlakha and Carl Kingsford. 2011. Network archaeology: uncovering ancient networks from present-day interactions. *PLoS Computational Biology* 7, 4 (2011), e1001119.

[12] Romualdo Pastor-Satorras, Eric Smith, and Ricard V Solé. 2003. Evolving protein interaction networks through gene duplication. *Journal of Theoretical Biology* 222, 2 (2003), 199–210.

[13] Satu Elisa Schaeffer. 2007. Graph clustering. *Computer Science Review* 1, 1 (2007), 27–64.

[14] Jithin K Sreedharan, Abram Magner, Ananth Grama, and Wojciech Szpankowski. 2019. Inferring temporal Information from a snapshot of a Dynamic Network. *Scientific reports* 9, 1 (2019), 3057.

[15] Jithin K Sreedharan, Krzysztof Turowski, and Wojciech Szpankowski. 2019. Revisiting Parameter Estimation in Biological Networks: Influence of Symmetries. (2019). https://doi.org/10.1101/674739 bioRxiv.

[16] Mansi Srivastava, Oleg Simakov, Jarrod Chapman, Bryony Fahey, Marie EA Gauthier, Therese Mitros, Gemma S Richards, Cecilia Conaco, Michael Dacre, Uffe Hellsten, et al. 2010. The Amphimedon queenslandica genome and the evolution of animal complexity. *Nature* 466, 7307 (2010), 720.

[17] Krzysztof Turowski and Wojciech Szpankowski. 2019. Towards Degree Distribution of Duplication Graph Models. (2019). https://www.cs.purdue.edu/homes/spa/papers/random19.pdf.

[18] Remco Van Der Hofstad. 2016. *Random graphs and complex networks*. Vol. 1. Cambridge University Press, Cambridge.