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

Krzysztof Turowski*, Jithin K. Sreedharan*, and Wojciech Szpankowski, *Fellow, IEEE*

***Abstract*—In *temporal ordered clustering*, given a single snapshot of a dynamic network in which nodes arrive at distinct time instants, 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 is of considerable significance in many applications ranging from tracking the expansion of fake news to mapping the spread of information. We first formulate our problem for a general dynamic graph, and propose an integer programming framework that finds the optimal clustering, 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). We then develop a sequential importance procedure and design unsupervised and semi-supervised algorithms to find temporal ordered clusters that efficiently approximate the optimal solution. To illustrate the techniques, we apply our methods to the vertex copying (duplication-divergence) model which exhibits some edge-case challenges in inferring the clusters as compared to other network models. Finally, we validate the performance of the proposed algorithms on synthetic and real-world networks.**

***Index Terms*—Clustering, dynamic networks, unsupervised learning, semi-supervised learning, temporal order**

I. 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 [1].

However, in dynamic networks that grow over time with nodes or edges getting added or deleted, the criterion of clustering based on its temporal characteristics finds significant relevance in practice since it helps us to study the existence of certain network structures and their future behavior. One approach to reason about the history of dynamic networks via clustering is guided by the problem of node labeling according to their arrival order when *only the structure* of the final snapshot of the network is provided. The availability of merely structure means that either we are given an unlabeled graph or the current node labels do not present any historical 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 [2], [3]. Figure 1 shows an example. In such cases, it is essential to classify nodes that are indistinguishable themselves in terms of arrival order into clusters $\{C_i\}$. Furthermore, the formed clusters also will be ordered as $C_1 \prec C_2 \prec \cdots$ so that for any $i < j$, all the nodes in the cluster C_i are estimated to arrive 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*.

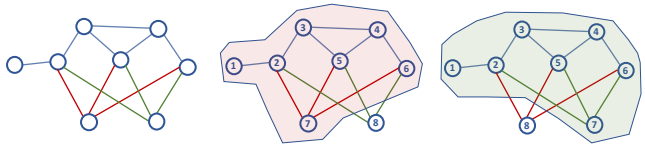


Fig. 1: Example showing how temporal clustering arises: a) the input graph without labels. b) and c) arbitrary labellings of arrival order with 1 representing the earliest arrival and 8 for the latest arrival. In b) and c), the last two arrived nodes 7 and 8 have the same set of neighbors. If we simulate the process of evolution starting from node 1, we observe that graphs in b) and c) at time 7 (i.e., with nodes $1-7$) are identical. Thus, nodes 7 and 8 in b) and c) are indistinguishable as to which arrived early between them (this observation holds for any labeling on the input graph), and the nodes behind these two labels are part of a temporal cluster.

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 rumor or epidemic networks, temporal ordered clustering can assist in identifying the sources and carriers of false information.

One of the major applications of temporal ordered clustering is in biological networks, especially protein-protein interaction (PPI) networks, as it is a difficult task to recover the history Our clustering identifies the evolution of biomolecules in the network and helps in predicting early proteins that are

K. Turowski is with the Theoretical Computer Science Department, Jagiellonian University, Krakow, Poland.
E-mail: krzysztof.szymon.turowski@gmail.com.
J. K. Sreedharan is with Wadhwani AI – AI for Social Good, Mumbai, Maharashtra 400093, India.
Email: jithin.k.s@gmail.com.
W. Szpankowski is with the Dept. of Computer Science and the NSF Center for Science and Information, Purdue University, West Lafayette, IN 47907, U.S.A.
E-mail: szpan@purdue.edu.
*Both the authors contributed equally to this research. This work was carried out when K. T. and J. K. S. were working at the NSF Center for Science of Information (CSoI), Purdue University.
This work was supported by NSF Center for Science of Information Grant CCF-0939370, and in addition by NSF Grants CCF-1524312, CCF-2006440, and CCF-2007238, National Science Center Poland, under Grant UMO-2016/21/B/ST6/03146 and Google Research Award.

Fig. 2: PPI network of Schizosaccharomyces pombe, fission yeast, with nodes colored and classified according to their estimated ancestral class, taxons (induced subgraph on 300 proteins). The clusters (taxons) are ordered according to their estimated phylogentic age as follows: Cellular Organisms Eukaryota Opisthokonta Ascomycota SCHPO.

known to be preferentially implicated in cancers and other diseases [4], [5], [6]. Understanding of the ancestral structure of interaction networks would immensely enhance the current research in discovering the processes behind the evolution of cellular systems. Moreover, recovery of the phylogenetic history of proteins in the PPI network reveals the functional modules. With our clustering scheme, we segregate the proteins into non-overlapping families of homologous proteins and the proteins in each cluster are believed to have descended from a common ancestral protein. Each of such clusters represents a taxon in the phylogenetic tree. An example of such recovery is shown in Figure 2. Here, the taxon of each protein and its phylogenetic age and order is estimated via a method outlined in [7]. Later in Section VI-B, we present the results of temporal ordered clustering on PPI networks of three species.

A. Our contributions

The main aim of our clustering formulation is to characterize its inherent limits via deriving the maximum accuracy that can be achieved for the number of comparable node pairs in the given cluster structure, and to develop algorithms for recovering the history of a dynamic network.

In Sections II and III we provide a general framework and derive an optimization problem for finding temporal ordered clusters in dynamic networks when only the final snapshot of its evolution is provided. Due to the high computational complexity involved in solving it, we reformulate the problem in terms of partial orders – for any node pair (D, E), a partial order f defines an order $D \prec_f E$ in which node D is specified to arrive earlier than node E. Such a partial order naturally translates into clusters of nodes and introduces an order among them.

Both the optimization problems depend on the knowledge of the probabilistic evolution of the graph model and the probability that any node D is older than any other node E, denoted as $?_{D-E}$. Therefore, in Section III-C we design a sequential importance sampling algorithm to estimate $?_{D-E}$ for any general graph model, and prove its convergence. The solution to a linear programming relaxation of the original optimization problem with coefficients as estimated $?_{D-E}$ presents an upper bound on the precision, a measure of the quality of temporal ordered clustering we define in Section II-B.

In Section IV, we propose approximate unsupervised solutions for temporal ordered clustering that are based on $?_{D-E}$ values estimated from the sequential importance sampling procedure, instead of solving the original optimization problems. Our experiments in Section VI-A show for small networks, these algorithms perform close to the theoretical bound outlined by the solution to the optimization problem. To further assess their performance, we also compare them to the algorithms based on certain properties of the network that are believed to be related heuristically to the age of the nodes (e.g. degrees, intersections of neighbor sets, etc).

In Section IV-B, we develop a semi-supervised technique to include any partial information, in the simplest case provided by a set of perfect pairs (i.e. pairs to be guaranteed to occur in the ordering), into the $?_{D-E}$-based algorithms. The experimental results show the improvement of the quality of solutions when the number of (random) perfect pairs provided as external knowledge.

In the second part of the paper (Sections V and VI), as an application of the proposed general technique, we focus on duplication-divergence or vertex copying dynamic network model (DD-model) [8] 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 to the remaining nodes (see Section V-A for details). The DD-model poses unique challenges for temporal ordered clustering in comparison with other graph models because of the features listed below:

– Non-equiprobable large number of permutations: In many of the graph models including the preferential attachment and Erdös-Rényi graph models, all the feasible permutations of the same structure representing node arrival orders are equally likely [2]. Later in the paper, we show with a counterexample that this is not the case in the DD-model. In other words, unlike in our previous work [9], we do not assume the isomorphic graphs that have positive probability under the graph model have the same probability. Moreover, in the DD-model, all the permutations of node labels with $=$ letters are valid unlike some models like the preferential attachment model; hence the effective space of total orderings is $=!$. Thus the DD-model stands as a corner case in the problem of node arrival order inference.
– Large number of symmetry: We provide evidence of a large number of automorphisms in a duplication-divergence graph, whereas it is known that in Erdös-Rényi

and preferential attachment graphs are asymmetric (when the automorphism group contains only the identity permutation) with high probability [2], [10].
– Ineffectiveness of degree-based techniques: In some models (including the preferential attachment model), the oldest nodes have larger expected degrees than the younger nodes over time, with high probability. But it is known that in the DD-model the average degree does not exhibit such a consistent trend [11], [12]. Thus any degree-based method is going to fail in the DD-model.

B. Related works and novelty in this work

This work covers the following range of topics.

1) Temporal or dynamic networks and graph models: In practice we encounter a lot of areas of scientific interest with networked data in which the structure of interactions varies over time [13], [14], [15], [16]. The pivotal influence of the temporal networks has seen in areas such as biochemical process (e.g., protein-protein interaction networks), epidemiology (e.g., disease spreading networks) or social sciences (e.g., friendship, human interaction or social influence networks) [17], [18], [19]

The representations of such temporal networks vary a lot. They are either described as graphs on a fixed number of nodes with connections between the nodes appearing or disappearing over time [20] or many times as growing networks with new nodes appear along with a batch of new edges are considered [21]. The theoretical models usually assume the second type, thus focus on the latter type of networks in this work to formulate the temporal ordered clustering problem and to provide some guarantees. The main example of network models is the preferential attachment model, but there exists a wide array of well-established models of such flavor [22], [23], [24], [25], [26]. Note that some models, e.g. [27], which are defined in terms of a fixed node set can be easily reinterpreted as networks with growing number of nodes.

Graph models, especially the growing networks with node additions, are expected to only represent the simple rules behind the evolution of real-world networks, yet as the graph size increases they represent many macroscopic and microscopic characteristics of real-world networks [28], [29], [30], [31], [32], [33]. For example, [20] mentions protein-protein interaction networks as one of the areas of application of temporal networks. The main mechanism behind the evolution of PPI networks, grounded at Ohno's hypothesis on genome growth and several empirical studies, is argued to be the duplication-divergence mechanism, which is a growing network paradigm [34], [35], [36].

We note here that growing graph models are studied in the literature, theoretically or experimentally, from the point of view of typical network measures: degree distribution, existence of power-law, centrality, occurrence of small subgraphs (motifs) [37], [38], [39], [40]. Our inference problem of temporal ordered clustering poses a different question from such previous works and deduces the latent information within the structure of an evolved dynamic graph about its evolution.

2) Clustering in static networks: Networks can be analyzed for patterns in two main ways: topological patterns and temporal patterns. The topological clustering of nodes is a classic problem in static networks. Its goal is to obtain a partition of the data set into sets such that the nodes in each set are similar or connected according to certain measures. In general, solutions to this problem follow two main approaches: 1) to define a similarity metric between node pairs, and choose clusters in a way that maximizes 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 [1].

3) Clustering in dynamic networks: 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 [41], [42], [43], [44]. However, this problem was exclusively concerned with the ("synchronic") connectivity or similarity of the network, even with added a time dimension, and it should not be confused with ("diachronic") clustering of events we consider in this paper. In the latter case we would like to group the events which occurred at a similar time or, equivalently, extract some information about the ordering of the events.

For dynamic graphs, the clustering problem has a natural interpretation as a study of the evolution of communities in its graph snapshots. Given a series of snapshots representing the graph structures over time, the goal here is to track substructures that are sufficiently close-knit according to some well-established distance measures or modularity (see [45] for an overview and comparison of static and dynamic clustering problems). New distance measures tailored to the clustering problem for dynamic graphs are studied in [46]. For clustering algorithms on dynamic graphs, [44] computed clusters for each snapshot and then evaluated matching between clusters for every pair of subsequent snapshots. Another technique in [42] extends spectral clustering on static networks to dynamic networks, obtaining clusterings over time. Since for large dynamic graphs with numerous snapshots this is infeasible, [47], [43] proposed heuristics which update clusters based only on the knowledge of differences (i.edge additions and deletions) between subsequent graph snapshots.

The temporal ordered clustering or partial order inference considered in this paper poses a very different problem in contrast to the classical dynamic clustering formulation. The optimization criterion for temporal ordered clustering introduces a fresh look taking into account the graph model and its actual temporal or evolutionary behavior (see Section III). The nodes inside our clusters 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.

4) Semi-supervised learning: The requirement that we have access only to a final state of the network might sound restrictive at first, but it is indeed reasonable in cases when the ground truth is not available e.g. for brain network. Since in general this condition might sound too restrictive at first,

however it is important to provide a way to include any additional information e.g. about precedence inferred from intermediary states of the network or from external sources. This way, we may change our problem into semi-supervised temporal ordered clustering

Previous works on semi-supervised clustering methods for data represented as vectors [48], [49] and their extensions to graphs [50] focus mainly on using the labeled nodes to de ne 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 coef cients of the associated linear programming (by restricting the sampling distribution of importance sampling, see Section IV-B)

5) Recovering history of dynamic networks: Node arrival order in the DD-model has been studied in [51] and [52], 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 their symmetries [2], [3]. Instead of total order, in this work we focus on deriving an optimal partial order of nodes of nodes (see Section II). Our methods are general and are applicable to a wide class of graph models, unlike our recent work [9] and [53] where the methods were speci c to the preferential attachment model and not extendable.

Compared to previous works, we introduce the following:

- Flexibility to choose between unsupervised and semi-supervised techniques: unuspervised when merely structure of the graph (unlabeled graph or the current node labels do not convey any historical information), and semi-supervised when age orderings between some node-pairs are provided.
- Optimization problem to nd theoretical limit of achievability and approximation algorithms for general growing graph models
- Extensive experiments on various synthetic and real-world networks (including PPI networks)

A preliminary version of this paper, containing only a shortened discussion of the partial ordering problem and unsupervised learning on synthetic graphs, has appeared before in [54].

II. Problem Formulation

Let G_t be the observed undirected and unweighted graph of t nodes with $V(G_t)$ being the set of vertices and $E(G_t)$ being the set of edges. The graph G_t is a result of evolution over time, starting from a seed graph G_{t_0} with t_0 nodes. At a time instant τ, when a new node appears, a set of new edges adjacent to the new node is added, and the graph will evolve into $G_{\tau+1}$. Since the change in graph structure occurs only when a new node is added, assuming the addition of a node as a time epoch, G_t also represents graph at time epoch t. The time epoch t_0 denotes the creation of the seed graph G_{t_0}[1].

[1]In the rest of the paper, we omit conditioning on the given G_{t_0} in all the expressions for the sake of brevity, if it is clear from the context.

Given only the snapshot of the dynamic graph G_t at time t, we usually do not know the time or order of arrivals of nodes. Essentially, our goal is to label each node with a number $\in \{1, \ldots, K\}$, such that all the nodes labeled k arrived before nodes with labels ℓ where $\ell > k$. The number of labels (clusters) K is unknown before and is a part of the optimal clustering formulation. 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 problem in the following way. Let G_t be a graph drawn from a dynamic random graph model $\mathcal{G}_t$ on t vertices in which nodes are labeled as $[t] = \{1, 2, \ldots, t\}$ according to their arrival, i.e., node v was the vth node to arrive. Let G_t evolve from the seed graph G_{t_0}. To model the lack of knowledge of the original labels, we subject the nodes to a permutation σ drawn uniformly at random from the symmetric group S_t on t letters ($\sigma \in S_t$), and we are given the graph $H_t := \sigma(G_t)$; that is, the nodes of G_t are randomly relabeled. We also use the notation $\mathcal{H}_t$ to denote the random graph behind H_t. Our original goal is to infer the arrival order in G_t after observing H_t, i.e., to nd σ^{-1}. The permutation σ^{-1} gives the true arrival order of the nodes of the given graph.

Instead of putting a constraint on recovering the whole permutation σ^{-1} or equivalently $\sigma(v) = v$ labels, we resort to strict (irre exive) partial orders. For a partial order $\prec$, a relation $u \prec v$ means that node u is older than node v according to the ordering $\prec$.

A. Relation between temporal ordered clusters and partial order set

Every partially ordered set can be represented by a clustering $\{C_k\}$ as follows. A strict partially ordered set can be represented initially by a directed acyclic graph (DAG) with nodes as the nodes in the graph G_t and directed edges as given by the partial order $\prec$: an edge from u to v exists when $u \prec v$. Then taking the transitive closure of this DAG will result in the DAG of the partial order set $\prec$. Now, all the nodes with in-degree 0 in the DAG will be part of cluster C_1 and the set of nodes with all the in-edges coming from nodes in C_1 will form cluster C_2. This process repeats until we get C_K. The number of clusters K is not de ned before but found from the DAG structure. Unlike the classical clustering, these clusters are ordered such that $C_1 \prec C_2 \prec \cdots \prec C_K$, where the relation $C_k \prec C_{k'}$, $k < k'$ is de ned as all the nodes inside the cluster C_k are estimated to be arrived earlier than all the nodes in the cluster $C_{k'}$, and all the nodes inside each cluster are considered to be identical in arrival order. We note here that not all partial orders result in a DAG that is weakly connected. If there are multiple components in the DAG corresponding to a partial order, each of them will give independent clustering. It might be due to the nodes in these separate components of the DAG are developed independently during evolution. Moreover, if there are nodes that are not part of any comparison in the partial order, we label them as unclassi ed.

In the following Section III-A, we formulate an optimization problem for the clusters and nd that the time complexity of its solution is t^5-times larger than that of the solution of the

optimization problem of partial orders in Section III-B. Hence in this paper, we focus only on the temporal-ordered clusters derived from the partial order.

We define an estimator q of the temporal ordered clustering[2] as a function q from the set of all labeled graphs on n vertices to the set of all partial orders on nodes $\{1, \dots, n\}$.

We consider estimators based on unsupervised and semi-supervised learning paradigms:

Unsupervised: In this case, the estimator does not have access to any information of the node arrival orders. Its results will be based only on the assumption that the graph model fits well the real-world network under consideration. In Section III we formulate an optimization problem for unsupervised learning and in Section IV we provide approximate solutions of the optimization.

Semi-supervised: In some of the real-world networks, partial information of the order of nodes is available - for some of the node pairs (u, v), it is revealed to the estimator that node u is arrived earlier than node v. Such node pairs are termed as *perfect pairs*. Taking this information into account would help the estimator that is initially based on fixed graph model to adapt to the real-data. The semi-supervised estimators introduced in Section IV learn the partial orders in the data without violating the perfect pairs.

B. Measures for evaluating partial order

For a partial order f, let $\Gamma(f)$ denote the number of pairs (u, v) that are comparable under f: i.e., $\Gamma(f) = |\{(u, v) : u \prec_f v\}|$, where $|\Gamma(f)| \geq \binom{n}{2}$.

Density: the density of a partial order is simply the number of comparable pairs, normalized by the total possible number, $\binom{n}{2}$. That is, $\chi(f) = \Gamma(f) / \binom{n}{2}$. Note that $\chi(f) \in [0, 1]$. Then the density of a partial order estimator q is simply its minimum possible density $\chi(q) = \min_\pi [\chi(q(\pi))]$.

Precision: it measures the expected fraction of correct pairs out of all pairs that are guessed by the partial order. That is

$$\psi(f) = \mathbb{E}\left[\frac{1}{\Gamma(f)} |\{(u, v) \in [n] : u \prec_f v \text{ and } \sigma^{-1}(u) < \sigma^{-1}(v)\}|\right].$$

For an estimator q, we also denote by $\psi(q)$ the quantity $\mathbb{E}[\psi(q(C(G_\pi)))]$. 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.

III. Solving the Optimization Problem

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

$$\psi(q) = \sum_\pi \Pr[C(G_\pi) = \pi] \frac{1}{\Gamma(q(\pi))} \mathbb{E}\left[|\{(u, v) \in [n] : u \prec_{q(\pi)} v \text{ and } \sigma^{-1}(u) < \sigma^{-1}(v)\}| \,\middle|\, C(G_\pi) = \pi\right].$$

Here σ and G_π 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 π, a partial order $q(\pi)$ that maximizes

$$\Psi_\chi(q) := \mathbb{E}\left[|\{(u, v) \in [n] : u \prec_{q(\pi)} v \text{ and } \sigma^{-1}(u) < \sigma^{-1}(v)\}| \,\middle|\, C(G_\pi) = \pi\right]$$

subject to the density constraint $\chi(q(\pi)) = \Gamma(q(\pi)) / \binom{n}{2} \geq \chi$, which says that we must have a certain minimum density of comparable pairs (here, $\chi \in [0, 1]$ is a parameter of the problem).

In the following first two subsections, we formulate the above optimization problem for two cases: when the estimator outputs the clusters and when it outputs the partial order. Each of these optimizations add a set of extra constraints to the original problem.

Let

$$p_{u \prec v}(\pi) := \Pr[\sigma^{-1}(u) < \sigma^{-1}(v) \,|\, C(G_\pi) = \pi] \quad (1)$$

be the probability that u is arrived before v given the relabeled graph π. The probability $p_{u \prec v}(\pi)$ turns out to be a critical quantity that serves as the coefficient in the linear programming approximations of the optimization problems and its estimation is explained in the last subsection of this section.[3]

A. Integer programming formulation for clusters

In this subsection, we restrict our optimization to linear cluster estimators, where the clusters are arranged in a total (linear) order.

To accomplish this optimization, we introduce, for each vertex v, a vector $\mathbf{x}_v = (x_{v,1}, \dots, x_{v,n})$, where $x_{v,k} = 1$ encodes the fact that node v is placed in cluster k.

Then Ψ_χ can be written in terms of integer programming (IP) formulation as

$$\sum_{u \neq v} \sum_{k=1}^{n} p_{u \prec v}(\pi) \frac{x_{u,k} x_{v,k}}{\sum_{i:\, i < k} \sum_{j=1}^{n} x_{i,j} x_{v,j}} \quad (2)$$

subject to the basic constraints[4]

$$\sum_{k=1}^{n} x_{v,k} = 1 \;\forall v \in [n] \quad \& \quad x_{v,k} \in \{0, 1\} \;\forall v, k \in [n].$$

We additionally have the following density constraint for a given χ:

$$\sum_{i:\, i < k} \sum_{j=1}^{n} x_{i,j} x_{v,j} \geq \chi \binom{n}{2}.$$

Each term of the form $x_{u,k} x_{v,k}$ becomes one only when the node u is classified into a cluster k that has lower precedence than node v's cluster k ($u \prec_{q(\pi)} v$). This corresponds to the event $u \prec_{q(\pi)} v$ with q as given by the clusters. The probability

[2] From now on, we use the terms node arrival order inferencing and temporal ordered clustering interchangeably in the paper

[3] We drop the dependence of π in $p_{u \prec v}(\pi)$ and $\%$ if it is clear from the context.

[4] Let the nodes in π take unique labels from the set $[n] = \{1, 2, \dots, n\}$. The original random graph G_π is assumed to be labeled from $[n]$, with label k indicating kth arrival node).

$?_{D-E}$ appears because of the event $\mathbb{1}(D \prec_{\hat{q}(\pi)} E)$ inside the expectation in ρ_Y. The denominator in (2) corresponds to $\mathbb{1}(D \preceq_{\hat{q}(\pi)} E)$.

That is, we have a quadratic rational integer program with linear basic constraints and a quadratic constraint introduced by the minimum density. We show now how to convert our program to a linear rational integer program with linear constraints.

We define new variables $I_{D-E-9} = G_{D-E-9}$, for $D-E \in \pi$ [9]. We can then eliminate the rational part of the integer program using the substitution

$$B = \left(\sum_{D-E \in \pi} I_{D-E-9} \right)^{-1} \quad \text{and} \quad I'_{D-E-9} = B\, I_{D-E-9}$$

With the above change of variables, the domain of I' is restricted to $[0, B]$. The density constraint

$$\sum_{D-E \in \pi} I_{D-E-9} \geq \frac{1}{2} \quad \Rightarrow \quad B \geq \frac{1}{2}.$$

Now we transform the integer program to a linear program by assuming I' takes continuous values with domain $[0, 1/2]$. We call the resulting optimization as LP-clusters.

Original integer program	LP approximation
$\max_{I} \dfrac{\sum_{D-E \in \pi} ?_{D-E}(\pi)\, I_{D-8-E-9}}{\sum_{F \prec_{\hat{q}(\pi)} F'} I_{F-:-F'-;}}$	$\max_{I'} \sum_{D-E \in \pi} ?_{D-E}(\pi)\, I'_{D-8-E-9}$
subject to	subject to
$I_{D-8-E-9} \in \{0, 1\}$	$I'_{D-8-E-9} \in [0, 1/2]$
$\forall D-8-E \in [9]$	$\forall D-8-E \in [9]$
$\sum_{D-E \in \pi} I_{D-8-E-9} \geq 2$	$\sum_{D-E \in \pi} I'_{D-8-E-9} \geq 1$
$\forall 8 \in [=]$, $\sum_{D-8-D=8} 1, \forall D \in [=]$	$\forall 8 \in [=]$, $\sum_{D-8-D-8} 1/2, \forall D \in [=]$
$I_{D-8-E=9} \leq I_{E-9-D-8}$	$I'_{D-8-E=9} \leq I'_{E-9-D-8}$
$\forall D-8-E \in [9]$	$\forall D-8-E \in [9]$
$I_{D-8-E=9} \leq I_{E-9-E-9}$	$I'_{D-8-E=9} \leq I'_{E-9-E-9}$
$\forall 8 \in [=]$, $\forall D-E-2 \in [9]$	$\forall 8 \in [=]$, $\forall D-E-2 \in [9]$

The first three constraints are direct translation of the constraints in (2), and the last two comes from the substitution $I_{D-8-E=9} \leq G_{D-8-E-9}$

Complexity analysis. The LP approximation presented above has $\Theta(=^4)$ decision variables and $\Theta(=^4)$ constraints. Therefore the complexity of solving this optimization, without taking into account the complexity of estimating $?_{D-E}$ will be of the order of $=^{12}$ in practice $\beta^2 2$ if 3 is the number of decision variables and 2 is the number of constraints [55, Section 1.2.2]).

The numerical experiments of the optimization in terms of clusters are presented later in Section VI-A. We also provide comparisons showing the formulation in terms of partial orders given in the next subsection computes much faster, yet outputs estimates with precision closer to that of cluster optimization.

B. Integer programming formulation for partial orders

In this subsection, we derive the optimal partial order among the nodes for the arrival order inference problem, extending some results from our recent work in [9].

We now represent the optimization problem with $\rho(q)$ as an integer program of partial order. For an estimator q, we define a binary variable H_{D-E} for each ordered pair $(D-E)$ as $H_{D-E} = 1$ when $D \prec_{\hat{q}(\pi)} E$. Note that $H_{D-E} = 0$ means either $D \succeq_{\hat{q}(\pi)} E$ or the pair $(D-E)$ is incomparable in the partial order $\hat{q}(\pi)$.

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 $\rho_Y(q)$. The constraints of the optimizations correspond to domain restriction, minimum density, and partial order constraints – antisymmetry and transitivity respectively. To use a linear programming approximation, we first convert the rational integer program into an equivalent truly integer program. With the substitution $B = 1 \Big/ \sum_{D-E \in \pi} H_{D-E}$ and $H'_{D-E} = B\, H_{D-E}$, the objective function is rewritten as a linear function of the normalized variables. These programs are equivalent if $H'_{D-E} \in \{0, B\}$, $B \geq 1/2$. For the LP relaxation, we assume H'_{D-E} as $[0, 1/2]$. We call the LP in this subsection as the LP-partial-order.

Original integer program	LP approximation
$\max_{H} \dfrac{\sum_{D-E \in \pi} ?_{D-E}(\pi)\, H_{D-E}}{\sum_{D-E \in \pi} H_{D-E}}$	$\max_{H'} \sum_{D-E \in \pi} ?_{D-E}(\pi)\, H'_{D-E}$
subject to	subject to
$H_{D-E} \in \{0, 1\}$, $\forall D-E \in [=]$	$H'_{D-E} \in [0, 1/2]$, $\forall D-E \in [=]$
$\sum_{D-E \in \pi} H_{D-E} \geq 1/2$	$\sum_{D-E \in \pi} H'_{D-E} \geq 1$
$H_{D-E} + H_{E-D} \leq 1$, $\forall D-E \in [=]$	$H'_{D-E} + H'_{E-D} \leq 1/2$, $\forall D-E \in [=]$
$H_{D-E} + H_{E-F} - H_{D-F} \leq 1$, $\forall D-E-F \in [=]$	$H'_{D-E} + H'_{E-F} - H'_{D-F} \leq 1/2$, $\forall D-E-F \in [=]$

The above integer program and LP-partial-order formulation is different from the LP-clusters in many ways. The idea of LP-partial-order is to relax the formulation of LP-clusters by focusing on the underlying partial order of clusters, rather than clusters itself. This simplifies the objective function, though it brings additional partial order constraints into the optimization. After finding the optimal partial order, we can derive the ordered clusters from it using the peeling technique in Section II-A. We note here that this may not need result in unique clusters. Many partial orders can have the same the cluster structure, especially when the DAG corresponding to the partial order contains multiple components.

The next lemma bounds the effect of approximating the coefficients $?_{D-E}$ on the optimal value of the integer program.

Lemma 1. Consider the integer program whose objective function is given by

$$\hat{\rho}_Y(q) = \frac{\sum_{D-E \in \pi} ?_{D-E}(\pi)\, H_{D-E}}{\sum_{D-E \in \pi} H_{D-E}}$$

with the same constraints as in the original integer program. Assume $\hat{\gamma}_{D-E}(n)$ can be approximated with $|\hat{\gamma}_{D-E}(n) - \gamma_{D-E}(n)| \le \epsilon$ uniformly for all $D-E$. Let q^* and $\hat{q}^*$ denote optimal points for the original and modified integer programs, respectively. Then $|\hat{Y}_\epsilon(\hat{q}^*) - Y(q^*)| \le 3\epsilon$ for arbitrary $\epsilon > 0$.

Proof. We extend the proof of [9, Lemma 5.1] – it now requires a weaker assumption $|\hat{\gamma}_{D-E}(n) - \gamma_{D-E}(n)| \le \epsilon$ instead of $|\hat{\gamma}_{D-E}(n) - \gamma_{D-E}(n)| \le \epsilon$ in [9].

Our goal is to upper bound

$$|Y(q^*) - \hat{Y}_\epsilon(\hat{q}^*)|.$$

We can rewrite this as

$$\begin{aligned}|Y(q^*) - \hat{Y}_\epsilon(q^*)| &+ |\hat{Y}_\epsilon(q^*) - Y(\hat{q}^*) + Y(\hat{q}^*) - \hat{Y}_\epsilon(\hat{q}^*)| \\ &\le |\hat{Y}_\epsilon(\hat{q}^*) - Y(\hat{q}^*)| + |\hat{Y}_\epsilon(q^*) - Y(q^*)| \\ &\quad + |\hat{Y}_\epsilon(q^*) - Y(\hat{q}^*)|\end{aligned}$$

Now, the first and second differences on the right-hand side are at most ϵ, since

$$|\hat{Y}_\epsilon(q^*) - Y(q^*)| \le \frac{\sum_{1 \le D \le E \le n} q_{D-E} |\hat{\gamma}_{D-E}(n) - \gamma_{D-E}(n)|}{\sum_{1 \le D \le E \le n} q_{D-E}} \le \epsilon \frac{\sum_{1 \le D \le E \le n} q_{D-E}}{\sum_{1 \le D \le E \le n} q_{D-E}} = \epsilon.$$

The remaining difference can be estimated as follows:

$$|\hat{Y}_\epsilon(q^*) - Y(\hat{q}^*)| \le |\hat{Y}_\epsilon(\hat{q}^*) - Y(\hat{q}^*)| \le \epsilon.$$

This inequality is a result of the fact that $\hat{q}^*$ is the optimal point for $\hat{Y}_\epsilon(\cdot)$ objective function. This shows that

$$|Y(q^*) - \hat{Y}_\epsilon(\hat{q}^*)| \le 3\epsilon$$

so we only incur a small additive error in the optimal precision by estimating the coefficients.

Complexity analysis and advantage over the cluster optimization. The LP approximation has $O(n^2)$ decision variables and $O(n^3)$ constraints appearing in the formulation. Thus computational complexity of the LP will be of the order of order of n^7 (without taking into account the estimation complexity of γ_{D-E}), which is much less than n^{12} complexity of cluster optimization in the previous subsection. Later in Section VI-A, we provide numerical comparisons showing the formulation in terms of partial orders computes much faster, yet outputs estimates with precision closer to that of cluster optimization.

C. Estimating coefficients using importance sampling

We now discuss the importance sampling approach to estimate the coefficient γ_{D-E} that is needed to solve the optimization problem. The following approach to estimate γ_{D-E} is applicable to any general graph model with Markovian evolution (conditioned on the present state of the graph, the new state is independent of the past state).

To estimate γ_{D-E} we classify dynamic graph models into two categories. Let $\Phi(G_n)$ be the set of all feasible permutations f which generates a positive probability graph G_n according to the distribution of the graph generation model.

i). Graph models with equiprobable isomorphic graphs. Here, two isomorphic graphs have same probability under the graph model. Formally, consider a graph $G_n^{(1)}$ with $P[G_n = G_n^{(1)}] > 0$ and another graph $G_n^{(2)}$, $G_n^{(2)} = f(G_n^{(1)})$ with $f \in \Phi(G_n^{(1)})$, then the equiprobable condition can be stated as $P[G_n = G_n^{(1)}] = P[G_n = G_n^{(2)}]$. Our previous work in [9] focus on such a case and derives the following result.

Lemma 2 ([9, Lemma 4.1 in Supplementary Information]) For all $E-F \in [n]$ and graphs G_n,

$$P[C^{-1}(E) \prec C^{-1}(F) | C(G_n) = G_n] = \frac{|f : f^{-1} \in \Phi(G_n), f^{-1}(E) \prec f^{-1}(F)|}{|\Phi(G_n)|}. \quad (3)$$

Though the graph models with such a property are not common, it include preferential attachment and Erdős-Rényi models. For preferential attachment model, we show in [9] that the estimation of right-hand side of (3) deduces to finding the proportion of linear extensions of a partial order (set of node pair orderings that hold with probability 1) satisfying $f^{-1}(E) \prec f^{-1}(F)$.

ii). Graph models with non-equiprobable isomorphic graphs. Many of the graph models do not possess equiprobable ismorphic graphs property. In this work, we propose a new estimation scheme based on importance sampling that is applicable to such a case for any general graph model with Markovian evolution.

We have, for $\gamma_{D-E} := P(C^{-1}(D) \prec C^{-1}(E) | C(G_n) = G_n)$,

$$\begin{aligned}\gamma_{D-E} &= \sum_{f : f^{-1} \in \Phi(G_n), f^{-1}(D) \prec f^{-1}(E)} P(C = f | C(G_n) = G_n) \\ &= \sum_{f : f^{-1} \in \Phi(G_n), f^{-1}(D) \prec f^{-1}(E)} \frac{P[C = f, C(G_n) = G_n]}{P[C(G_n) = G_n]} \\ &= \sum_{f : f^{-1} \in \Phi(G_n), f^{-1}(D) \prec f^{-1}(E)} \frac{P[G_n = f^{-1}(G_n)] P[C = f]}{\sum_{f^{-1} \in \Phi(G_n)} P[G_n = f^{-1}(G_n)] P[C = f]} \\ &= \frac{\sum_{f : f^{-1} \in \Phi(G_n), f^{-1}(D) \prec f^{-1}(E)} P[G_n = f^{-1}(G_n)]}{\sum_{f^{-1} \in \Phi(G_n)} P[G_n = f^{-1}(G_n)]}, \quad (4)\end{aligned}$$

where we used the fact that $P[C = f] = 1/n!$ since it is independent of G_n.

We now derive an estimator for γ_{D-E} by approximating the numerator and denominator of right-hand side in (4). The γ_{D-E} expression involves summing over permutations from the feasible set, i.e. $f^{-1} \in \Phi(G_n)$ ($f^{-1}(G_n)$ gives a positive probable graph by the definition of $\Phi(G_n)$). Since there are at most $n!$ permutations to check for feasibility, direct sampling from $\Phi(G_n)$ is impossible in many cases. However, we remark that each permutation $f^{-1} \in \Phi(G_n)$ invokes a chain structure when the graph has a Markovian evolution, as follows. Applying f^{-1} to G_n is essentially relabeling of nodes in G_n from $[n]$. Then starting from labeling a guess of the youngest node with n, by reverse engineering the Markovian evolution of the graph, to find the node with label $B \prec n$ we need to know only the node $B+1$ and the graph G_{B+1}. Based on this observation, to

estimate the denominator in right-hand side of (4) we propose a sequential importance sampling strategy in Theorem 1 that generalizes to any localized sampling distribution (probability to choose node B after selecting node $B+1$) which meets a certain criterion. This is directly extendable to estimating the numerator in (4) too by putting an extra restriction to the sampled permutation. Later Lemma 3 presents an estimator of $?_{D-E}$ using the technique derived in Theorem 1.

Let $R_t = +(t)$ denote the set of candidates for youngest nodes at time t. The set R_t depends on the graph model. For example, in case of preferential attachment model, in which a new node attaches s edges to the existing nodes with a probability distribution proportional to the degree of the existing node, R_t is the set of $<$-degree nodes. We consider only permutations that do not change the initial graph G_{t_0} labels. For instance, 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 $_{t_0}$ as $_{t_0}$ itself. Since we assume G_{t_0} is known, $?_{D-E}$ expression in (4) has an additional conditioning on G_{t_0}.

Let $X(t-L)$ represent the graph in which the node $L2 R_t$ is deleted from G_t. Then the graph sequence $H_t = G_t-H_{t-1} = X(t-L)-\cdots H_{t_0} = G_{t_0}$ forms a nonhomogeneous Markov chain – nonhomogeneous because the state space of $\{H_B\}_{B=t_0}$ changes with B and thus the transition probabilities too. Similarly $G_t-G_{t-1}-\cdots-G_{t_0}$ also make a Markov chain, and for a fixed permutation f, $f(t) = t$, both the above Markov chains have same transition probabilities. Let us also define the posterior probability of producing t from $X(t-L)$ as

$$F(X(t-L)-t) := P[H_t = tjH_{t-1} = X(t-L)]\bullet \quad (5)$$

The following theorem characterizes our estimator. For a Markov chain, let E_G denote the expectation with starting state G. Let G_B be the set of all labeled graphs on B vertices.

Theorem 1 (Sequential importance sampling). Consider a time-nonhomogeneous Markov chain $H_t = tH_{t-1} = X(t-l_t)-\bullet\bullet,\bullet$ where $l_t 2 R_t-l_{t-1} 2 R_{t-1}-\cdots 4$ be the nodes removed randomly by the Markov chain and let its transition probability matrices be $\{\&_B = [@_B(0-00)]g_{B=t_0}$ for any two graphs $0 2 G_B$ and $00 2 G_{B-1}$. Then we have

$$\sum_{f^{-1}2(t)} P[G_t = f^{-1}(t)j\ _{t_0}] = E_{H_t=t}\left[\prod_{B=t_0+1}^{t}\frac{F(X(B-l_B)-l_B)}{@_B(B-X(B-l_B))}\right]\bullet$$

Proof. Now we have the following iterative expression for the denominator of $?_{D-E}$

$$?^{denom}_{D-E}(t-t_0) := \sum_{f^{-1}2(t)} P[G_t = f^{-1}(t)j\ _{t_0}] \quad (6)$$

$$= \sum_{l_t 2 R_t}\sum_{f^{-1}2(t)} P[G_t = f^{-1}(t)-G_{t-1} = f_1^{-1}(X(t-l_t))j\ _{t_0}]-$$

where $f_1 2 (t-1$ is the permutation f with "l_t maps to=" removed. Now we can rewrite the above expression as

$$\sum_{l_t 2 R_t}\sum_{f^{-1}2(t)} P[G_t = f^{-1}(t)jG_{t-1} = f_1^{-1}(X(t-l_t))]$$

$$P[G_{t-1} = f_1^{-1}(X(t-l_t))j\ _{t_0}]\bullet$$

Note that $P[G_t = f^{-1}(t)jG_{t-1} = f_1^{-1}(X(t-l_t))]$ for a fixed f (thus f_1) is equivalent to $F(X(t-l_t)-t)$. Now introducing a transition probability $f\&_B = [@_B(8-9)]g_{B=t_0}$ for the Markov chain $f\ _B g_{B=t_0}$, and using importance sampling,

$$?^{denom}_{D-E}(t-t_0) = \sum_{l_t 2 R_t}\frac{F(X(t-l_t)-t)}{@_t(t-X(t-l_t))}@_t(t-X(t-l_t))$$

$$\sum_{f^{-1}2(X(t-l_t))} P[G_{t-1} = f^{-1}(X(t-l_t))j\ _{t_0}]\bullet$$

$$= \sum_{l_t 2 R_t}\frac{F(X(t-l_t)-t)}{@_t(t-X(t-l_t))}@_t(t-X(t-l_t))$$

$$?^{denom}_{D-E}(X(t-l_t)-t_0)- \quad (7)$$

with $?^{denom}_{D-E}(t_0-t_0) = 1$. Here $@_t(t-X(t-l_t))$ is the transition probability to jump from $H_t = t$ to $H_{t-1} = X(t-l_t)$.

Now let $`(t-t_0) = E_{H_t=t}\left[\prod_{B=t_0+1}^{t}\frac{F(X(B-l_B)-l_B)}{@_B(B-X(B-l_B))}\right]$

Then we have,

$$`(t-t_0) = \sum_{l_t 2 R_t}\frac{F(X(t-l_t)-t)}{@_t(t-X(t-l_t))}@_t(t-X(t-l_t))$$

$$E_{H_{t-1}=X(t-l_t)}\left[\prod_{B=t_0+1}^{t}\frac{F(X(B-l_B)-l_B)}{@_B(B-X(B-l_B))}\right] \quad (8)$$

$$= \sum_{l_t 2 R_t}\frac{F(X(t-l_t)-t)}{@_t(t-X(t-l_t))}@_t(t-X(t-l_t))$$

$$`(X(t-l_t)-t_0)- \quad (9)$$

where (8) follows from the Markov property.

Defining the function at t_0 as

$$\frac{F(X(t_0-l_{t_0})-t_{0+1})}{@_{t_0}(t_{0+1}-X(t_0-l_{t_0}))} = 1- \text{ for any } l_{t_0}$$

we note here that the iteration (9) of $`(t-t_0)$ is identical to that of $?^{denom}_{D-E}$ in (7). This completes the proof.

Remark 1. Note that unlike $@_B(B-X(B-l_B))$, which is under our control to design a Markov chain, $F(X(B-l_B)-l_B)$ is a well-defined fixed quantity (see (14)). The only constraint for the transition probability matrices $\{\&_B g_{B=t_0}$ is that it should be chosen to be in agreement with the graph evolution such that the choices of jumps from $_B$ to $_{B-1}$ restricts to removing nodes from R_B, and it depends on the graph model.

p_{D-E} estimator. Now we can form the estimator for $?_{D-E}$ for a node pair $(D-E)$ as follows. Let $^{(i)}$ be the vector denoting the sampled node sequence of the ith run of the Markov chain. It can either represent a vector notation as $^{(i)} = (l^{(i)}_t-l^{(i)}_{t-1}-\cdots-l^{(i)}_{t_0+1})$ or take a function form $^{(i)}(B)$ denoting the new label of a vertex B in G_t. We propose the following estimator and show that it has asymptotic consistency.

Lemma 3 (Estimator and its consistency). Let the estimator of ?$_{D-E}$ for all $D-E \in$ $_=$, formed from : samples of the sequential importance sampling (see Theorem 1) be

$$\hat{?}^{(:)}_{D-E} = \frac{\sum_{8=1}^{:} 1_{f\circledR^{(8)}(D) \ddot{Y} \circledR^{(8)}(E) g} \prod_{B=}^{=_0+1} \frac{F(X(_B \circledR^{(8)}_B) - _B)}{@_B(_B - X(_B \circledR^{(8)}_B))}}{\sum_{8=1}^{:} \prod_{B=}^{=_0+1} \frac{F(X(_B \circledR^{(8)}_B) - _B)}{@_B(_B - X(_B \circledR^{(8)}_B))}} \bullet \quad (10)$$

Then $\hat{?}^{(:)}_{D-E} !$?$_{D-E}$ a.s. as : ! 1 .

Proof. Using Theorem 1 and based on the observation that the Markov sample paths in different runs are independent and identically distributed, the numerator and denominator in the right-hand side of (10) converge separately to that of (4) by strong law of large numbers (in almost surely sense). Then by invoking continuous mapping theorem, we can prove that their ratio also converges to ?$_{D-E}$ almost surely.

Theorem 1 and Lemma 3 provide us the flexibility and convenience to sample the permutations and estimate ?$_{D-E}$ via a wide-range of sampling distributions. In the next section, we consider two such candidate distributions.

IV. 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 ?$_{D-E}$ according to (4) is of exponential complexity. With Theorem 1 and eq. (10), we can approximate ?$_{D-E}$ as the empirical average of Markov chain based sample paths. We try two different importance sampling distributions f@$_B$g$_{B=}$:

local-unif-sampling with transition probabilities

$$@_B(_B - X(_B - I_B)) = \frac{1}{jR_Bj} \bullet \quad (11)$$

high-prob-sampling forms the Markov chain with

$$@_B(_B - X(_B - I_B)) = \frac{F(X(_B - I_B) - _B)}{\sum_{D \in R_B} F(X(_B - D) - _B)} \bullet \quad (12)$$

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 VI-A, it has much slower rate of convergence than local-unif-sampling.

Moreover both implementations differ significantly in their computational complexity: at each step B high-prob-sampling requires \$(=^{2}) computations – as there are \$(=) possibilities for immediate ancestor of I$_B$ in X($_B$ - I$_B$) which is needed for calculating the posterior probability F and there are \$(=) possibilities for the sum in the denominator. On the other hand local-unif-sampling requires only \$(=) – counting jR$_B$j by checking all the nodes. In some graph models (like the DD-model in Section V-A), all the nodes in $_B$ can be part of R$_B$ with a positive probability, and local-unif-sampling will essentially become uniform sampling.

The local-unif-sampling can be further improved with the acceptance-rejection sampling technique: at a step C, randomly sample a node D from +($_C$) (instead of sampling from R$_C$). Then calculate the probability that the node D be the youngest node in the graph. If this probability is positive, we accept D as +$_C$ and if it is zero, we randomly sample again from +($_C$).

However, note that both high-prob-sampling and local-unif-sampling have net running times \$(=^{3}:) and \$(=^{2}:) respectively, with : being the number of sample paths computed by our sequential importance sampling algorithm. In practice this means that these algorithms can be applied only to small- and medium-sized networks (up to e.g. 10^4 nodes), but not to large ones unless someone has access to really large computational power.

Now we assume that ?$_{D-E}$ are estimated for all D and E to propose algorithms for temporal clustering. In fact, according to Lemma 1, we only need to have max$_{D-E}$ j?$_{D-E}$?$_{D-E}$j _ for a small _ ¡ 0. Thus for small ?$_{D-E}$, ?$_{D-E}$ can be assumed to be zero. We can then use LP-partial-order in Section III-B with the estimated ?$_{D-E}$ as the coefficients. Due to the huge computational complexity associated with the LP solution, we now propose the following unsupervised and semi-supervised approximation algorithms based on the estimates of ?$_{D-E}$.

A. Unsupervised solution

sort-by-?$_{D-E}$-sum algorithm. For this algorithm, we construct a new complete graph with the node set same as that of $_=$ and edge weights as ?$_{D-E}$. Let us now define a metric ?$_D$:= $\sum_{E \in 2+}$?$_{D-E}$ for every node D of $_=$. Since ?$_{D-E}$ denotes the probability that node D is older than node E, ?$_D$ would give a high score when a node D becomes the oldest node. Our ranking is then sorted order of the ?$_D$ values.

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

?$_{D-E}$-threshold algorithm. Here, each of the estimated ?$_{D-E}$'s is compared against a threshold g. Only the node pairs that are strictly greater than this condition are put into the estimator output partial order. Note that if g = 0•5, we get a total order in virtually all the relevant cases – as the only possible tie occurs when ?$_{D-E}$ = ?$_{E-D}$ = 0•5.

B. Semi-supervised solution

Suppose we have partial true data of node arrival order available from some external sources. Let it be ordered in partial order as f$_{orig}$ = f(D-E)g, in which for the pair (D-E), D is the older than E. Let f$_{train}$ f$_{orig}$ be the training set and the let the test set be f$_{test}$:= f$_{orig}$nf$_{train}$. Let jf$_{train}$j = Ujf$_{orig}$j for some 0 Ÿ U Ÿ 1. With the knowledge of f$_{train}$, we modify the estimation of ?$_{D-E}$ as follows. The set of removable nodes R$_B$ at each instant B is modified to R$_B$nNR$_B$, where NR$_B$

Algorithm 1 Temporal Ordered Clustering: Semi-supervised

Input: graph G_t, graph model $G_t = \cdots$ description, training set of partial orders f_{train}, number of sample paths $\cdots$
Output: Clusters $C_1\ C_2 \cdots C_\cdots$

1: procedure TEMPORALORDEREDCLUSTERING
2: for $\cdots$ from 1 to $\cdots$ do
3: for B from t down to t_0 do
4: Find R_B and NR_B by (13)
5: $R_B \leftarrow R_B \cup NR_B$
6: Sample $I_B^{(\cdots)}$ using a sampling method local-unif-sampling (11) or high-prob-sampling (12)
7: end for
8: end for
9: Estimate $\hat{\cdots}^{(\cdots)}_{D-E}$ using (4)
10: Use algorithm sort-by-$\hat{\cdots}_{D-E}$ sum or $\hat{\cdots}_{D-E}$ threshold to estimate clusters of nodes $C_1, C_2, \cdots, C_\cdots$
11: return $C_1, C_2, \cdots, C_\cdots$
12: end procedure

Fig. 3: Semi-supervised learning example DAG for $f_{train} = \{(D, E), (E, F), (F, G), (H, F)\}$: $NR_B = \{E, F, G\}$ and $R_B = \{D, H\}$.

is the set of nodes that can not be included in the removable nodes as it would violate the partial order for f_{train}. It is defined as follows:

$$NR_B := \{D : (D, E) \in f_{train} \wedge D \notin \{I_B, \ldots, I_t\}\} \cup \{I_t = I_{t_0}\}. \quad (13)$$

After estimating $\hat{\cdots}_{D-E}$ with the redefined R_B, we employ sort-by-$\hat{\cdots}_{D-E}$ sum algorithm or $\hat{\cdots}_{D-E}$ threshold algorithms to find partial order. An example of R_B construction is shown in Figure 3.

Algorithm 1 summarizes our semi-supervised algorithm. The algorithm will become unsupervised when there is no f_{train} and step-5 is removed.

V. Temporal Ordered Clustering for Duplication-Divergence Model

A. Duplication-divergence model (DD-model)

We consider definition of the DD-model by Pastor-Satorras et al. [56]. Given an undirected, simple seed graph G_{t_0} on t_0 nodes and target number of nodes t, the graph G_{t+1} with $t+1$ nodes[5] evolves from the G_t as follows: first, a new vertex E is added to G_t. Then the following steps are carried out:

Duplication: Select a node D from G_t uniformly at random. The node E then makes connections to $N(D)$, the neighbor set of D.

Divergence: Each of the newly made connections from E to $N(D)$ are deleted with probability $1-\cdots$. Furthermore, for all the nodes in G_t to which E is not connected, create an edge from it to E independently with probability $\frac{\cdots}{t}$.

The above process is repeated until the number of nodes in the graph is equal to t. We denote the graph G_t generated from the DD-model with parameters $\cdots$ and $\cdots$, starting from seed graph G_{t_0}, by $G_t = \text{DD-model}(t, \cdots, G_{t_0})$.

The posterior probability $P(X(I_B, I_t) \mid I_B)$, which is defined in (5) and used in Theorem 1 and high-prob-sampling, can be calculated for the DD-model as follows. For a node $I_B \in R_B$, we say a node D is its parent if D can be selected from the graph $X(I_B, I_t)$ for the duplication step when I_B is added into $X(I_B, I_t)$. The probability of having the node D as the parent of $I_B \in R_B$ in the DD-model is

$$\begin{aligned} &P(X(I_B, I_t) \mid D \mid I_B) \\ &= \frac{1}{B-1}\cdots^{|N(I_B)\setminus N(D)|}(1-\cdots)^{|N(D)\cap N(I_B)|} \\ &\quad \frac{\cdots}{B-1}^{|N(I_B)\cap N(D)|}\left(1-\frac{\cdots}{B-1}\right)^{(B-1)-|N(I_B)\cup N(D)|}. \end{aligned} \quad (14)$$

The above expression can be inferred directly from the definition of the DD-model as follows. We first pick D as a parent node of I_B with probability $\frac{1}{B-1}$. Then to calculate the probabilities of edge addition events retrospectively, we observe that edges from I_B to the nodes in the $N(I_B) \setminus N(D)$ stayed with probability $\cdots$, but edges to $N(D) \cap N(I_B)$ were dismissed with probability $1-\cdots$. We also have to take into account the edges between I_B and vertices outside of $N(D) \cup \{D\}$ – each were chosen independently with probability $\frac{\cdots}{B-1}$ and they are exactly the edges from I_B to $N(I_B) \cap N(D)$.

Now $P(X(I_B, I_t) \mid I_B) = \sum_{D \in P_B(I_B)} P(X(I_B, I_t) \mid D \mid I_B)$, where $P_B(I_B)$ represents possible parents of I_B in R_B.

Since all permutations have positive probability in this version of the model, we have $R_B = \{\cdots(B)\}$ and $\cdots(B) = B$.

B. Greedy algorithms for clustering

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

sort-by-degree. The nodes are sorted by the degree and arranged into clusters $\{\cdots_0, \cdots_1, \ldots\}$. Cluster $\cdots_1$ contains nodes with the largest degree.

[5] The subscript $\cdots$ with $G_\cdots$ can also be interpreted as time instant

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 (D, E) such that $N(D)$ contains $N(E)$. This condition holds when $A = 0$. When $A > 0$, we consider $|N(E) \setminus N(D)| \le A$ as A is the average number of extra connections a node makes apart from duplication process. In most real-world data, we estimate A to be smaller than 1, and hence the original check is sufficient.

peel-by-neighborhood . Here, we find the set $\{D : \forall E \in N(E) \setminus N(D) | \le A\}$ (as mentioned before, it is sufficient to check $N(E) \setminus N(D)$ 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 t_0. This algorithm makes use of the DAG of the neighborhood relationship and includes isolated nodes into the bins.

C. 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-Rényi graphs and preferential attachment graphs.

First, for a fixed graph G_t on t vertices, let us consider a set of graphs $\mathrm{Adj}(G_t) = \{f(G_t) : f \in \mathrm{Aut}(G_t)\}$. It is obvious that for the Erdős-Rényi model, any graph in $\mathrm{Adj}(G_t)$ is generated equally likely with a given seed graph G_{t_0}. Such property was also proved for the preferential attachment model in [2]. However, this does not hold for DD-model graphs as shown in the following example.

Fig. 4: Example of asymmetric graph

For the graph $G_t^{(1)}$ presented in Figure 4, let G_{t_0} consists of vertices 1, 2, 3, and let the parameters of the DD-model be $p = 0.2$ and $A = 0$. The $P[G_t = G_t^{(1)}]$ can be calculated iteratively using (14) as 0.068. Now, consider the permutation

$$f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 5 & 4 \end{pmatrix}.$$

Then $f \in \mathrm{Aut}(G_t^{(1)})$. Let $G_t^{(2)} = f(G_t^{(1)})$. The $P[G_t = G_t^{(2)}]$ is 0.051, and conditioned on the same structure probabilities of $G_t^{(1)}$ and $G_t^{(2)}$ are 0.5744 and 0.4256 respectively.

Second, it is well known that both the Erdős-Rényi graphs and preferential attachment graphs are asymmetric[6] with high probability [10], [2]. 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. 5. This is in accordance with many real-world networks (see Table III for examples).

Fig. 5: $E \log |\mathrm{Aut}(G_t)|$, $G_t \sim$ DD-model $(2000, p, A = 20)$, where $|\mathrm{Aut}(G_t)|$ is the number of automorphisms in graph G_t.

Last, the behavior of the degree at time C of the node arrived at an earlier time B (denoted by $\deg_C(B)$) is different for all three models. For Erdős-Rényi graph with edge probability p, it is known that $E[\deg_C(B)] = p(C-1)$. For the preferential attachment graphs $E[\deg_C(B)] = \sqrt{C B}$ ([57], Theorem 8.2). However, for the DD-model $E[\deg_C(B)] = {}^{1}C\,B^{p}\,B^{2p-1}$ for any $C \gg B$ [11]. Note that when $B = \Theta(1)$ – the case of very old nodes – the average degree is of order C^p. For $B = C$, we have $E[\deg_C(C)] = \Theta(C^{2p-1})$ which is growing only for $p > 1/2$. For example, when $p = 1$ degrees of all the nodes on average are of order $\Theta(C)$. 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. Moreover, Frieze et al. [12] has shown that $\deg_C(B)$ is concentrated around the mean for $B = \Theta(1)$ in the sense that for any $\alpha > 1$ we observe polynomial tail:

$$\Pr[\deg_C(B) > \alpha E[\deg_C(B)] \log C] = \Pr[\deg_C(B) < E[\deg_C(B)] / \log C] = \Theta(C^{-\beta}),$$

for certain fixed constant α and a constant β dependent on α. However, this is not the case for the last vertices, since they are copied from already existing nodes in the network, which would explain the ineffectiveness of greedy degree-based heuristics for $p \le 1/2$.

VI. Experiments

In this section, we evaluate our methods on synthetic and real-world data sets. We made publicly available all the code and data of this project at https://github.com/krzysztof-turowski/duplication-divergence.

We present the following results in the coming sections.

Synthetic networks:

[6] An automorphism or symmetry of a graph is an isomorphism from a graph to itself. We say that G is symmetric if it has at least one nontrivial symmetry and that G is asymmetric if the only symmetry of G is the identity permutation.

– How well the LP-partial-order performs in comparison with the LP-clusters? (Figures 6 and 7)
– Fixing the LP-partial-order, how is the convergence of $?_{D-E}$'s that are estimated via sequential importance sampling schemes *local-unif-sampling* and *high-prob-sampling* as to the exact $?_{D-E}$ in terms of resulting precision? (Figure 8)
– Fixing LP-partial-order for the LP formulation and *local-unif-sampling* for the importance sampling strategy, we study the performance of unsupervised algorithms in comparison with greedy strategies specific to the DD-model. (Figure 9)
– For the semi-supervised algorithms, we show results (precision and density) for various parameter configurations and study their influence on the performance. (Tables I and II)

Real-world networks: For the semi-supervised algorithms how the precision improves with a small change in the training size, and how does the results compare against greedy algorithms of the DD-model? (Figure 10 and Table IV)

Maximum likelihood estimation: For deriving total order, a natural solution will be the maximum likelihood estimator (MLE).

$$\arg\max_{f \in \Omega(\mathcal{G})} P[G_n = \mathbf{C}^{-1}(\mathcal{G}) \mid \mathbf{C}^{-1} = f]$$

But we do not consider MLE explicitly here because it is known that many networks exhibit large number of symmetries (see Table III for some examples), and thus there will be large number of total orders that achieve the MLE criterion with low value of precision. In fact, our optimal formulation in Section III already captures the MLE solutions and outputs them if they have high precision. Moreover for general graph models, the MLE computation would require checking all $\Omega(\mathcal{G})$ which incurs $\Omega(n!)$ computational complexity.

A. Synthetic networks

In the following results on synthetic networks, k_{tries} denote the number of Markov chain sample paths (for sequential importance sampling) used for estimating $?_{D-E}$ for all $\{D-E \in \Omega(\mathcal{G})\}$. All the studies are performed on multiple graph realizations from the DD-model with specified parameters, and the results are averaged over them. When we make a comparison based on LP formulation, we plot precision $Y = (X \cap Y)/X$ vs minimum density $Y = (X \cap Y)/Y$ in accordance with the formulations in Sections III-A and III-B.

In Figure 6, we compare the performance of the linear programming approximations LP-cluster (Section III-A) and LP-partial-order (Section III-B). Since clustering output from the LP-cluster scheme induces a partial order, we use the same measures of precision and density that are defined for partial order for comparing performances of LP-cluster and LP-partial-order schemes. Our experiments confirm that for the same graph, with the same set for $\{?_{D-E} = \{D-E \in \Omega(\mathcal{G})\}$, the performance of them are nearly identical. However, Figure 7 shows that the difference between the running time of both the formulations is huge – the LP-clusters which finds clusters becomes barely feasible, whereas LP-partial-order which outputs partial order runs in a reasonable time.

Fig. 6: Comparison between LP-clusters and LP-partial-order formulation: $n = 30$ DD-model ($\alpha = 0.6$ – $A = 1.0$ – $\epsilon_0 = 10$) and $k_{\text{tries}} = 100{,}000$. Results are averaged over 100 graph generations. Sampling method: *local-unif-sampling*.

Fig. 7: Time plot for LP-partial-order vs. LP-clusters. All the experiments were performed on 48-CPU cluster, with Intel(R) Xeon(R) CPU E7-8857 v2 @ 3.00GHz and 256GB RAM.

Figure 8 examines the precision of LP-partial-order obtained with approximated $\{?_{D-E} \mid D-E \in \Omega(\mathcal{G})\}$ via sequential importance strategies (*local-unif-sampling* and *high-prob-sampling*) and that obtained with the $\{?_{D-E} \mid D-E \in \Omega(\mathcal{G})\}$ that is calculated exactly by considering all the possible $n!$ orderings. We consider a small size ($n = 13$) example here since it becomes infeasible to compute the exact curve for larger values of n. 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. Thus, along with the computational reasons stated in Section IV, we use *local-unif-sampling* in the subsequent experiments.

In Figure 9, we compare results of the unsupervised algorithms with the estimated optimal curve via *local-unif-sampling*. It turns out that greedy algorithms (sort-by-

Fig. 8: Results on synthetic networks with exact curve: DD-model (13– ?1•0– $_{=0}$) for ? = 0•3 (left) and 0•6 (right), averaged over 100 graphs. $_{=0}$ is generated from Erdős-Rényi graph with = $_0$ = 4 and ?$_0$ = 0•6; precision\ vs minimum density Y (X Y)

Fig. 9: Results on synthetic networks with greedy and unsupervised learning ?$_{D-E}$-based algorithms: $_{=}$ DD-model (50– ?1•0– $_{=0}$) for ? = 0•3 (left) and 0•6 (right), averaged over 100 graphs. ?$_{D-E}$-based algorithms use f_{tries} = 100 000. $_{=0}$ is generated from Erdős-Rényi model with = $_0$ = 10 and ?$_0$ = 0•6. The theoretical curve is estimated via local-unif-sampling ; precision\ vs minimum density Y (X Y)

degree , sort-by-neighborhood , peel-by-degree and peel-by-neighborhood) perform reasonably well for small ?, but their performance deteriorates for higher values of ?. On the other hand, ?$_{D-E}$-based algorithms (sort-by- ?$_{D-E}$ sum and ?$_{D-E}$ threshold) offer consistent, close to the theoretical bound, behavior different values of ? (figure shows only two ?s due to space limitations). Moreover, the bin size j j in sort-by- ?$_{D-E}$ sum and threshold g in ?$_{D-E}$ threshold algorithm offer a trade-off between higher precision and higher density. The larger the bin size or the higher the threshold, we observe a decrease in density, but increase in precision as we stay close to the theoretical curve.

Table I contains the results of semi-supervised learning extensions of the ?$_{D-E}$-based algorithms. A small increase in the percentage of the training set yields a large increases in precision for all sets of parameters. Moreover, for larger bin size in sort-by- ?$_{D-E}$ sum algorithm we observe mainly only an increase in precision with U, but for ?$_{D-E}$ threshold algorithm both X and \ grow visibly with U, especially for large g. In turn, when we x U and increase the bin size in sort-by- ?$_{D-E}$ sum algorithm, precision remains almost same, but density decreases signi cantly. And if we do the analogous procedure for ?$_{D-E}$ threshold algorithm (but U and increase threshold), then precision grows, but in expense of a visible fall of density. All the above conclusions are summarized in Table II.

		? = 0•3		? = 0•6	
Algorithm	U	X	\	X	\
sort-by- ?$_{D-E}$sum , j j= 1	0•001	1•0	0•598	1•0	0•613
sort-by- ?$_{D-E}$sum , j j= 1	0•01	1•0	0•643	1•0	0•650
sort-by- ?$_{D-E}$sum , j j= 1	0•1	1•0	0•836	1•0	0•832
sort-by- ?$_{D-E}$sum , j j= 10	0•001	0•769	0•605	0•769	0•626
sort-by- ?$_{D-E}$sum , j j= 10	0•01	0•768	0•661	0•767	0•660
sort-by- ?$_{D-E}$sum , j j= 10	0•1	0•758	0•864	0•759	0•859
?$_{D-E}$threshold , g = 0•5	0•001	1•0	0•604	1•0	0•617
?$_{D-E}$threshold , g = 0•5	0•01	1•0	0•637	1•0	0•649
?$_{D-E}$threshold , g = 0•5	0•1	1•0	0•829	1•0	0•823
?$_{D-E}$threshold , g = 0•9	0•001	0•010	0•906	0•028	0•871
?$_{D-E}$threshold , g = 0•9	0•01	0•020	0•951	0•090	0•907
?$_{D-E}$threshold , g = 0•9	0•1	0•521	0•966	0•559	0•960

TABLE I: Results on synthetic networks with semi-supervised learning ?$_{D-E}$based algorithms: = DD-model (50– ?1•0– $_{=0}$), averaged over 100 graphs. ?$_{D-E}$based algorithms use $_{tries}$ = 100–000. $_{=0}$ is Erdos-Renyi graph with = 10 and ?$_0$ = 0•6.

Algorithm	Fixed	Free	Free	Free
sort-by- ?$_{D-E}$sum	U	j j%	X &	\
sort-by- ?$_{D-E}$sum	j j	U %	X	\ %
?$_{D-E}$threshold	U	g %	X &	\ %
?$_{D-E}$threshold	g	U %	X %	\ %

TABLE II: Conclusions from synthetic data: how the metrics behave by xing one of the parameters and keeping other free. The symbol indicates the changes are not signi cant.

B. Real-world networks

We consider the following real-world networks which have the ground truth (or an estimated knowledge) of node and edge age arrival order available. The directed networks are treated as undirected in our studies. The rst three datasets are taken from SNAP repository [58] and the protein-protein interaction data is collected from the BioGRID[7].

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. In this network, many nodes share the same arrival time and date, and hence the true arrival order of nodes is available only in bins of count 1-457.

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. It shows the evolution of hyperlinks between articles of the Simple English Wikipedia.

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 to E indicates used sent a private message to use at time C Number of nodes is 1-899 and number of edges is 59-835.

Protein-protein interaction (PPI) networks. We consider the following PPI networks of three species. In each network, the nodes represent proteins and an edge is present if the incident proteins are found to be interacting. The networks formed from PPI data are further cleaned by removing self-interactions (self-loops), multiple interactions (multiple edges), and interspecies (organisms) interactions of proteins. Thus the considered PPI networks only have physical and intra-species interactions

- Mus Musculus (House mouse) 9-849 nodes and 18-380 edges.
- Saccharomyces cerevisiae (Baker's yeast) 6-252 nodes and 531-400 edges.
- Schizosaccharomyces pombe (Fission yeast) 4-177 nodes and 58-084 edges.

Here since the ground truth of protein phylogenetic ages (taxon ages) and orders are not available, they are estimated as follows. It is reasonable to expect that the same protein which appeared over different species also appears in their common ancestor. Hence proteins shared across many different, distant species are supposed to be older than others. More precisely, the age of a protein is based on a family's appearance on a species tree, and it is estimated via protein family databases and ancestral history reconstruction algorithms. We use Princeton Protein Orthology Database (PPOD) [59] along with OrthoMCL [60] and PANTHER [61] for the protein family database and asymmetric Wagner parsimony as the ancestral history reconstruction algorithm. These algorithms can be accessed via ProteinHistorian software [7].

Table III shows estimated parameters of the duplication-divergence model for the above networks using the tting technique from [62].

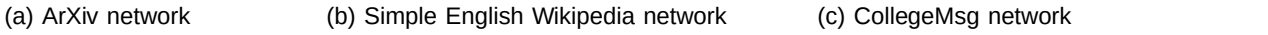
Network ($_{obs}$)	log|Aut($_{obs}$)|		
ArXiv	12•59	0•72	1•0
Wikipedia	1018•94	0•66	0•5
CollegeMsg	231•54	0•65	0•45
House mouse	7827•17	0•96	0•32
Baker's yeast	266•99	0•98	0•35
Fission yeast	674•90	0•983	0•85

TABLE III: Parameters of the duplication-divergence model estimated for the real-world networks considered in this paper.

Figures 10 and 11 show the result of semi-supervised learning. Here U represents the proportion of all pairs that is considered as training set, i.e., size of the training set is U $\frac{=}{2}$. We randomly pick U $\frac{=}{2}$ pairs and the results presented are average over 100 different such random sets. We observe that a small increase in U 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 where data is often assumed to be independent. This helps us to get a near-perfect clustering (precision close to 1) with only 1% of the labeled nodes.

Finally, the semi-supervised approach helps to obtain a signi cant improvement over the greedy algorithms. Unsupervised algorithms, implied from very small U in Figures 10 and 11, have precision 2 [0•5–0•6], which is only marginally

[7] https://thebiogrid.org



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

Fig. 10: Real-world networks: results of semi-supervised learning on social networks; precision vs proportion of considered node pairs U

better than random guess. However, as it is shown in Table IV, greedy algorithms find orderings with precision ranging from 0.47 to 0.63 for significant values of the density (it's easy to achieve a precision of 0.78 in CollegeMsg dataset when the density of pairs outputted is as low as 0.01) that is also close to random guess. Our semi-supervised algorithms, with a small change in U, significantly outperforms greedy algorithms.

VII. Discussion and future work

In this article we presented a framework for clustering nodes in dynamic networks based on latent temporal information. We provided a way to find the maximum achievable precision (as a measure of the quality of clustering), and proposed several algorithms that perform well and that are capable of including some external information about the precedence of vertices in their arrival to the network.

Further work in our proposed framework can go in several directions. For example, one can explore various ways to speed up the algorithms presented in this work, as even $\Theta(n^2 \cdot \ell)$ time complexity for a network on n vertices with ℓ sampled paths turned out to be a significant obstacle for applying them to large real-world networks. The improvement can be accomplished, for example, by finding a good importance sampling distribution, which will lead to faster convergence of our algorithms. From a theoretical perspective, there remains an interesting question of finding bounds on the convergence speed of estimates of $\mathbb{P}_{D-E}$ (probability that node D is arrived earlier than node E) with various importance sampling distributions. One can also look into clever bookkeeping techniques which will result in reducing the computation time of a single path in our sequential importance sampling algorithm. Another direction is the application of the proposed framework and solution to other types of random network models that not only involve the addition of vertices and edges, but also deletion of them.

References

[1] S. E. Schaeffer, "Graph clustering," Computer Science Review, vol. 1, no. 1, pp. 27–64, 2007. 1, 3

[2] T. Łuczak, A. Magner, and W. Szpankowski, "Asymmetry and structural information in preferential attachment graphs," Random Structures and Algorithms, pp. 1–23, 2019. 1, 2, 3, 4, 11

[3] K. Turowski, A. Magner, and W. Szpankowski, "Compression of Dynamic Graphs Generated by a Duplication Model," 56th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2018, Monticello, IL, USA, October 2-5, 2018, 2018, pp. 1089–1096. 1, 4

[4] M. Srivastava, O. Simakov, J. Chapman, B. Fahey, M. Gauthier, T. Mitros, G. Richards, C. Conaco, M. Dacre, U. Hellsten et al., "The amphimedon queenslandica genome and the evolution of animal complexity," Nature, vol. 466, no. 7307, p. 720, 2010. 2

[5] T. Ideker and R. Sharan, "Protein networks in disease," Genome research, vol. 18, no. 4, pp. 644–652, 2008. 2

[6] F. E. Faisal and T. Milenković, "Dynamic networks reveal key players in aging," Bioinformatics, vol. 30, no. 12, pp. 1721–1729, 2014. 2

[7] J. Capra, A. Williams, and K. Pollard, "Proteinhistorian: tools for the comparative analysis of eukaryote protein origin," PLoS Computational Biology, vol. 8, no. 6, p. e1002567, 2012. 2, 14

[8] R. Solé, R. Pastor-Satorras, E. Smith, and T. Kepler, "A model of large-scale proteome evolution," Advances in Complex Systems, vol. 5, no. 01, pp. 43–54, 2002. 2

[9] J. K. Sreedharan, A. Magner, A. Grama, and W. Szpankowski, "Inferring temporal information from a snapshot of a dynamic network," Scientific Reports, vol. 9, no. 1, p. 3057, 2019. 2, 4, 6, 7

[10] J. H. Kim, B. Sudakov, and V. Vu, "On the asymmetry of random regular graphs and random graphs," Random Structures & Algorithms, vol. 21, no. 3-4, pp. 216–224, 2002. 3, 11

[11] K. Turowski and W. Szpankowski, "Towards degree distribution of duplication graph models," 2019, https://www.cs.purdue.edu/homes/spa/papers/random19.pdf. 3, 11

[12] A. Frieze, K. Turowski, and W. Szpankowski, "Degree distribution for duplication-divergence graphs: Large deviations," Graph-Theoretic Concepts in Computer Science - 46th International Workshop, WG 2020, Leeds, UK, June 24-26, 2020, Revised Selected Papers, ser. Lecture Notes in Computer Science, I. Adler and H. Müller, Eds., vol. 12301. Springer, 2020, pp. 226–237. 3, 11

[13] G. Caldarelli, Scale-free networks: complex webs in nature and technology. Oxford University Press, 2007. 3

[14] R. Cohen and S. Havlin, Complex networks: structure, robustness and function. Cambridge University Press, 2010. 3

[15] B. Gonçalves and N. Perra, Social phenomena: From data analysis to models. Springer, 2015. 3

[16] A.-L. Barabásiet al., Network science. Cambridge university press, 2016. 3

[17] A. Li, S. Cornelius, Y.-Y. Liu, L. Wang, and A.-L. Barabási, "The fundamental advantages of temporal networks," Science, vol. 358, no. 6366, pp. 1042–1046, 2017. 3

[18] P. Holme and J. Saramäki, "Temporal networks," 2013. 3

[19] ——, Temporal Network Theory. Springer, 2019. 3

[20] ——, "Temporal networks," Physics Reports, vol. 519, no. 3, pp. 97–125, 2012. 3

[21] P. Holme, "Modern temporal network theory: a colloquium," The European Physical Journal B, vol. 88, no. 9, p. 234, 2015. 3

[22] F. Chung, L. Lu, T. G. Dewey, and D. Galas, "Duplication models for biological networks," Journal of Computational Biology, vol. 10, no. 5, pp. 677–687, 2003. 3

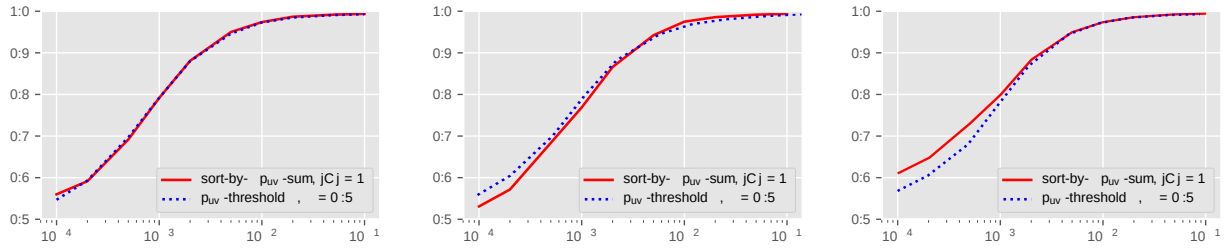
[23] I. Ispolatov, P. Krapivsky, and A. Yuryev, "Duplication-divergence model of protein interaction network," Physical Review E, vol. 71, no. 6, p. 061911, 2005. 3

[24] C. Wiuf, M. Brameier, O. Hagberg, and M. Stumpf, "A likelihood approach to analysis of network data," Proceedings of the National Academy of Sciences, vol. 103, no. 20, pp. 7566–7570, 2006. 3

[25] S. Boccaletti, D.-U. Hwang, and V. Latora, "Growing hierarchical scale-free networks by means of nonhierarchical processes," International Journal of Bifurcation and Chaos, vol. 17, no. 07, pp. 2447–2452, 2007. 3

	ArXiv		Wikipedia		CollegeMsg		House mouse		Baker's yeast		Fission yeast	
Greedy algorithm	δ	θ	δ	θ	δ	θ	δ	θ	δ	θ	δ	θ
`sort-by-degree`	0.98	0.47	0.96	0.59	0.92	0.63	0.99	0.51	0.78	0.50	0.96	0.51
`peel-by-degree`	0.98	0.46	0.96	0.59	0.92	0.63	0.99	0.51	0.79	0.50	0.96	0.51
`sort-by-neighborhood`	0.0001	0.51	0.03	0.60	0.01	0.78	0.006	0.56	0.08	0.51	0.03	0.47
`peel-by-neighborhood`	0.13	0.50	0.80	0.593	0.75	0.61	0.66	0.52	0.77	0.50	0.86	0.50

TABLE IV: Real-world networks: results of greedy algorithms; density δ vs precision θ



(a) House mouse (b) Baker's yeast (c) Fission yeast

Fig. 11: Real-world networks: results of semi-supervised learning on PPI networks; precision θ vs proportion of considered node pairs α

[26] C. Cooper and A. Frieze, "A general model of web graphs," *Random Structures & Algorithms*, vol. 22, no. 3, pp. 311–335, 2003. 3

[27] N. Perra, B. Gonçalves, R. Pastor-Satorras, and A. Vespignani, "Activity driven modeling of time varying networks," *Scientific Reports*, vol. 2, p. 469, 2012. 3

[28] M. Medo, "Statistical validation of high-dimensional models of growing networks," *Physical Review E*, vol. 89, no. 3, p. 032801, 2014. 3

[29] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins, "Microscopic evolution of social networks," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 462–470. 3

[30] S. Teichmann and M. M. Babu, "Gene regulatory network growth by duplication," *Nature Genetics*, vol. 36, no. 5, pp. 492–496, 2004. 3

[31] J. Berg, M. Lässig, and A. Wagner, "Structure and evolution of protein interaction networks: a statistical model for link dynamics and gene duplications," *BMC Evolutionary Biology*, vol. 4, no. 1, p. 51, 2004. 3

[32] F. Hormozdiari, P. Berenbrink, N. Pržulj, and S. C. Sahinalp, "Not all scale-free networks are born equal: the role of the seed graph in PPI network evolution," *PLoS Computational Biology*, vol. 3, no. 7, p. e118, 2007. 3

[33] R. Colak, F. Hormozdiari, F. Moser, A. Schönhuth, J. Holman, M. Ester, and S. C. Sahinalp, "Dense graphlet statistics of protein interaction and random networks," in *Biocomputing 2009*. Singapore: World Scientific Publishing, 2009, pp. 178–189. 3

[34] S. Ohno, *Evolution by gene duplication*. Springer Science & Business Media, 2013. 3

[35] J. Zhang, "Evolution by gene duplication: an update," *Trends in ecology & evolution*, vol. 18, no. 6, pp. 292–298, 2003. 3

[36] M. Shao, Y. Yang, J. Guan, and S. Zhou, "Choosing appropriate models for protein–protein interaction networks: a comparison study," *Briefings in bioinformatics*, vol. 15, no. 5, pp. 823–838, 2014. 3

[37] P. Krapivsky and B. Derrida, "Universal properties of growing networks," *Physica A: Statistical Mechanics and its Applications*, vol. 340, no. 4, pp. 714–724, 2004. 3

[38] G. Bebek, P. Berenbrink, C. Cooper, T. Friedetzky, J. H. Nadeau, and S. C. Sahinalp, "Improved duplication models for proteome network evolution," in *Systems Biology and Regulatory Genomics*. Springer, 2007, pp. 119–137. 3

[39] K. Evlampiev and H. Isambert, "Conservation and topology of protein interaction networks under duplication-divergence evolution," *Proceedings of the National Academy of Sciences*, vol. 105, no. 29, pp. 9863–9868, 2008. 3

[40] R. Lambiotte, P. Krapivsky, U. Bhat, and S. Redner, "Structural transitions in densifying networks," *Physical Review Letters*, vol. 117, no. 21, p. 218301, 2016. 3

[41] A. Loukas and P. Vandergheynst, "Spectrally approximating large graphs with smaller graphs," in *International Conference on Machine Learning*, Stockholm, Sweden, 2018, pp. 3243–3252. 3

[42] F. Liu, D. Choi, L. Xie, and K. Roeder, "Global spectral clustering in dynamic networks," *Proceedings of the National Academy of Sciences*, vol. 115, no. 5, pp. 927–932, 2018. 3

[43] R. Görke, P. Maillard, C. Staudt, and D. Wagner, "Modularity-driven clustering of dynamic graphs," in *International Symposium on Experimental Algorithms*. Berlin, Heidelberg: Springer, 2010, pp. 436–448. 3

[44] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks," in *2010 International Conference on Advances in Social Networks Analysis and Mining*. Washington, DC, USA: IEEE, 2010, pp. 176–183. 3

[45] P. Bedi and C. Sharma, "Community detection in social networks," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 6, no. 3, pp. 115–135, 2016. 3

[46] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng, "Evolutionary spectral clustering by incorporating temporal smoothness," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007, pp. 153–162. 3

[47] N. P. Nguyen, T. N. Dinh, Y. Shen, and M. T. Thai, "Dynamic social community detection and its applications," *PloS one*, vol. 9, no. 4, p. e91431, 2014. 3

[48] E. Bair, "Semi-supervised clustering methods," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 5, no. 5, pp. 349–361, 2013. 4

[49] S. Basu, A. Banerjee, and R. Mooney, "Semi-supervised clustering by seeding," in *International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 27–34. 4

[50] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semi-supervised graph clustering: a kernel approach," *Machine Learning*, vol. 74, no. 1, pp. 1–22, 2009. 4

[51] S. Li, K. P. Choi, T. Wu, and L. Zhang, "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)*, vol. 10, no. 6, pp. 1412–1421, 2013. 4

[52] S. Navlakha and C. Kingsford, "Network archaeology: uncovering ancient networks from present-day interactions," *PLoS Computational Biology*, vol. 7, no. 4, p. e1001119, 2011. 4

[53] J.-G. Young, G. St-Onge, E. Laurence, C. Murphy, L. Hébert-Dufresne, and P. Desrosiers, "Phase transition in the recoverability of network history," *Physical Review X*, vol. 9, no. 4, p. 041056, 2019. 4

[54] K. Turowski, J. K. Sreedharan, and W. Szpankowski, "Temporal ordered clustering in dynamic networks," 2020, to appear in Proceedings of IEEE International Symposium on Information Theory. 4

[55] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004. 6

[56] R. Pastor-Satorras, E. Smith, and R. V. Solé, "Evolving protein interaction networks through gene duplication," *Journal of Theoretical Biology*, vol. 222, no. 2, pp. 199–210, 2003. 10