Temporal Ordered Clustering in Dynamic Networks

Krzysztof Turowski*
Jagiellonian University, Poland
Email: krzysztof.szymon.turowski@gmail.com

Jithin K. Sreedharan*
Purdue University, U.S.A.
Email: jithinks@purdue.edu

Wojciech Szpankowski
Purdue University, U.S.A.
Email: szpan@purdue.edu

***Abstract*—Given a single snapshot of a dynamic network in which nodes arrived at distinct time instants along with edges, we aim at inferring a partial order σ between the node pairs such that $u <_\sigma v$ indicates node u arrived earlier than node v in the graph. The inferred partial order can be deduced to a natural clustering of the nodes into K ordered clusters $C_1 \prec \cdots \prec C_K$ such that for $i < j$, nodes in cluster C_i joined the network before nodes in cluster C_j, with K being a data-driven parameter and not known upfront. We first formulate our problem for a general dynamic graph, and propose an integer programming framework that finds the optimal partial order, 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 design algorithms to find temporal ordered clusters that efficiently approximate the optimal solution. To illustrate our techniques, we apply our methods to the vertex copying model (also known as the duplication-divergence model).**

I. Introduction

Dynamic networks grow over time with nodes or edges getting added or deleted. Understanding temporal characteristics of such networks is significant in practice since it helps us to study the existence of certain network structures and their future behaviour. One approach to reason about the history of dynamic networks 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 [1], [2]. In such cases, we can find a partial order σ between the node pairs such that $u <_\sigma v$ indicates node u arrived earlier than node v in the graph. Such a partial order naturally translates into clusters of nodes $\{C_i\}$ and introduces an order among the clusters 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 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*.

Temporal ordered clustering or partial order 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 [3]. In rumor or epidemic networks, temporal ordered clustering can assist in identifying the sources and carriers of false information.

*These authors contributed equally to this work.

Our contributions.

- We provide a general framework and derive an optimization problem for finding partial order of nodes (according to their arrival order) in dynamic networks when only the final snapshot of its evolution is provided. The optimization problem depends on the knowledge of the probabilistic evolution of the graph model and the probability that any node u is older than any other node v, denoted as p_{uv}. We then design a sequential *importance sampling* algorithm to estimate p_{uv} 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 p_{uv}, presents an upper bound on the clustering quality.
- In the second part 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 to the remaining nodes (see Section V 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 Erdős-Rényi graph models, all the feasible permutations of the same structure representing node arrival order are equally likely [1]. This is not the case in the DD-model (a counter example is given in the extended version of this paper [4] due to space limitations). In other words, unlike in our previous work [5], we do not assume the isomorphic graphs that have positive probability under graph model have the same probability. Moreover, in the DD-model, all permutations on n letters are valid unlike some models like preferential attachment model and hence the effective space of total orderings is $n!$.
  - *Large number of symmetry*: A duplication-divergence graph contains a large number of automorphisms (see Fig. 4 in the extended version [4]), whereas it is known that Erdős-Rényi and preferential attachment graphs are

asymmetric with high probability [6], [1].

– *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 larger expected degrees than the youngest 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 [4], [7]. Thus any method based on degrees is bound to fail here.

Prior related work. 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 [8], [9], [10], [11]. The temporal ordered clustering or partial order inference 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 III). Incidentally, the main aim of our clustering is to characterize the inherent limits of recovering the history of a dynamic network. 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.

Node arrival order in the DD-model has been studied in [12] and [13], 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 [1], [2]. 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 [5] where the methods were specific to the preferential attachment model and not extendable.

II. Problem Formulation

Let H_n be the observed undirected and unweighted graph of n nodes with $V(H_n)$ being the set of vertices and $E(H_n)$ being the set of edges. The graph H_n is a result of evolution over time, starting from a seed graph H_{n_0} with n_0 nodes. At a time instant k, when a new node appears, a set of new edges adjacent to the new node is added, and 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. The time epoch n_0 denotes the creation of the seed graph G_{n_0}.

Given only the snapshot of the dynamic graph H_n at time n, we usually do not know the time or order of arrivals of nodes. Essentially, 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 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_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 evolve from the seed graph G_{n_0}. 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 S_n, and we are given the graph $H_n := \pi(G_n)$; that is, the nodes of G_n are randomly *relabeled*. We also use the notation $\mathcal{H}_n$ to denote the random graph behind H_n. 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.

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

Relation between temporal ordered clusters and partial order set. Every partially ordered set can be represented by a clustering ($\{C_i\}$) 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 H_n and directed edges as given by the partial order 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. Unlike the classical 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. 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 unclassified.

We define an estimator ϕ of the temporal ordered clustering as a function $\phi : \mathbb{G}_n \to \mathbb{S}_n$, where $\mathbb{G}_n$ is the set of all graphs on n vertices and $\mathbb{S}_n$ is the set of all partial orders on nodes $1, \ldots, n$.

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, $\delta(\sigma) = K(\sigma)/\binom{n}{2}$. Note that $\delta(\sigma) \in [0, 1]$. Then the density of a partial order estimator ϕ is simply its minimum

possible density $\delta(\phi) = \min_{H_n}[\delta(\phi(H_n))]$.

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

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

For an estimator ϕ, we also denote by $\theta(\phi)$ the quantity $\mathbb{E}[\theta(\phi(\boldsymbol{\pi}(\mathcal{G}_n)))]$.

III. Solving the Optimization Problem

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

$$\theta(\phi) = \sum_{H_n} \Pr[\boldsymbol{\pi}(\mathcal{G}_n) = H_n]\frac{1}{K(\phi(H_n))}$$
$$\times\, \mathbb{E}\left[|\{u,v \in [n]\colon u <_{\phi(H_n)} v, \boldsymbol{\pi}^{-1}(u) < \boldsymbol{\pi}^{-1}(v)\}|\middle|\boldsymbol{\pi}(\mathcal{G}_n) = H_n\right].$$

Here $\boldsymbol{\pi}$ and $\mathcal{G}_n$ 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_n, a partial order $\phi(H_n)$ that maximizes

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

subject to the density constraint $\delta(\phi(H_n)) = K(\phi(H_n))/\binom{n}{2} \geq \varepsilon$.

A. Integer programming formulation

In this subsection we extend some results from our recent work in [5]. 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 $u <_{\phi(H_n)} v$. Note that $x_{u,v} = 0$ means either $u >_{\phi(H_n)} v$ or the pair (u,v) is incomparable in the partial order $\phi(H_n)$. Let $p_{u,v}(H_n) = \Pr[\boldsymbol{\pi}^{-1}(u) < \boldsymbol{\pi}^{-1}(v)|\boldsymbol{\pi}(\mathcal{G}_n) = H_n]$ is the probability that u arrived before v given the relabeled graph H_n.

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 is rewritten as a linear function of the normalized variables. These programs are equivalent if $y_{u,v} \in \{0, s\}$, $s = 1/\varepsilon\binom{n}{2}$. For the LP relaxation, we assume $y_{u,v}$ as $[0, s]$.

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* $\hat{J}_{\varepsilon,\lambda}(\phi) = \frac{\sum_{1\leq u<v\leq n} \hat{p}_{u,v}(H_n)x_{u,v}}{\sum_{1\leq u\neq v\leq n} x_{u,v}}$*, with the same constraints as in the original IP. Assume* $p_{u,v}(H_n)$ *can be approximated with* $|\hat{p}_{u,v}(H_n) - p_{u,v}(H_n)| \leq \lambda$ *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* $\lambda > 0$*.*

Original integer program	**LP approximation**
$\max_x \frac{\sum_{1\leq u\neq v\leq n} p_{u,v}(H_n)x_{u,v}}{\sum_{1\leq u\neq v\leq n} x_{u,v}}$	$\max_y \sum_{1\leq u\neq v\leq n} p_{u,v}(H_n)y_{u,v}$
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 proof of the above lemma is an extension of [5, Lemma 5.1, Supplementary Material] – we require a weaker assumption $|\hat{p}_{u,v}(H_n) - p_{u,v}(H_n)| \leq \lambda$ instead of $|\hat{p}_{u,v}(H_n)/p_{u,v}(H_n) - 1| \leq \lambda$ in [5].

B. Estimating coefficients with importance sampling

We now discuss the importance sampling approach to estimate p_{uv} that is needed to solve the optimization problem. The following approach to estimate p_{uv} 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) unlike our previous work in [5] which is specific to preferential attachment graphs.

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

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

With a way to approximate the numerator and denominator of (1), we now derive an estimator $p_{u,v}$. The basic idea is to sample feasible permutations from the structure of H_n. But since sampling from actual graph distribution incurs huge complexity, in what follows, we provide a method to use any sampling distribution that satisfies certain constraints. Let $\mathcal{R}_{H_n} \subseteq V(H_n)$ denote the set of *candidates* for youngest nodes at time n, and let $\mathcal{P}_{H_n}(v)$, represents possible parents of v in H_n, i.e., the nodes v may select for duplication. The set $\mathcal{R}_{H_n}$ depends on the graph model. For example, 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, $\mathcal{R}_{H_n}$ is the set of m-degree nodes. We consider only permutations that do not change the initial graph G_{n_0} labels. Thus we define H_{n_0} as G_{n_0} itself. Since we assume H_{n_0} is known, $p_{u,v}$ expression in (1) has an additional conditioning of H_{n_0}.

Let $\delta(H_n, v_n)$ represent the graph in which node v_n is deleted from H_n, where $v_n \in \mathcal{R}_{H_n}$. Then the graph sequence

$\mathcal{H}_n = H_n, \mathcal{H}_{n-1} = \delta(H_n, v_n), \ldots, \mathcal{H}_{n_0} = H_{n_0}$ forms a non-homogeneous Markov chain – nonhomogeneous because the state space $\{\mathbb{H}_s\}_{s \le t}$ changes with s and thus the transition probabilities too. Similarly $\mathcal{G}_n, \mathcal{G}_{n-1}, \mathcal{G}_{n_0}$ also make a Markov chain, and for a fixed permutation σ, $\sigma(G_n) = H_n$, both the above Markov chains have same transition probabilities. Let us also define the posterior probability of producing H_n from $\delta(H_n, v_n)$ as $w(\delta(H_n, v_n), H_n) := \mathbb{P}[\mathcal{H}_n = H_n | \mathcal{H}_{t-1} = \delta(H_n, v_n)]$

The following theorem characterizes our estimator. See [4] for the proof. For a Markov chain, let $\mathbb{E}_x$ denote the expectation with starting state x.

Theorem 1 (Sequential importance sampling)**.** *Consider a time-nonhomogeneous Markov chain $\mathcal{H}_n = H_n, \mathcal{H}_{n-1} = \delta(H_n, \mathcal{V}_n), \ldots$, where $\mathcal{V}_n \in \mathcal{R}_{H_n}, \mathcal{V}_{n-1} \in \mathcal{R}_{H_{n-1}}, \ldots$ be the nodes removed randomly by the Markov chain and let its transition probability matrices be $\{Q_s = [q_s(F', F'')]\}_{s \le t}$ for any two graphs $F' \in \mathbb{G}_s$ and $F'' \in \mathbb{G}_{s-1}$. Then we have*

$$\sum_{\sigma^{-1} \in \Gamma(H_n)} \mathbb{P}[\mathcal{G}_n = \sigma^{-1}(H_n) | H_{n_0}] = \mathbb{E}_{\mathcal{H}_n = H_n} \left[\prod_{s \le n}^{n_0+1} \frac{w(\delta(H_s, \mathcal{V}_s), H_s)}{q_s(H_s, \delta(H_s, \mathcal{V}_s))} \right].$$

Note that unlike $q_s(H_s, \delta(H_s, v_s))$, which is under our control to design a Markov chain, $w(\delta(H_s, v_s), H_s)$ is a well-defined fixed quantity (see (3)). The only constraint for the transition probability matrices $\{Q_s\}_{s \le n}$ 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 $\mathcal{R}_{H_s}$.

***p_{uv}* estimator.** 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 $\boldsymbol{v}^{(k)} = (v_n^{(k)}, v_{n-1}^{(k)}, \ldots, v_{n_0+1}^{(k)})$ or take a function form $\boldsymbol{v}^{(k)}(s)$ denoting the new label of a vertex s in H_n. The estimator $\hat{p}_{s,t}^{(k)}$ is now, for all $s, t \in H_n$

$$\hat{p}_{s,t}^{(k)} = \frac{\sum_{i=1}^{k} \mathbf{1}_{\{\boldsymbol{v}^{(i)}(s) < \boldsymbol{v}^{(i)}(t)\}} \prod_{s \le n}^{n_0+1} \frac{w(\delta(H_s, \boldsymbol{v}_s^{(i)}), H_s)}{q_s(H_s, \delta(H_s, \boldsymbol{v}_s^{(i)}))}}{\sum_{i=1}^{k} \prod_{s \le n}^{n_0+1} \frac{w(\delta(H_s, \boldsymbol{v}_s^{(i)}), H_s)}{q_s(H_s, \delta(H_s, \boldsymbol{v}_s^{(i)}))}}. \quad (2)$$

Since the Markov sample paths are independent, using Theorem 1, strong law of large numbers and continuous mapping theorem, we can prove that $\hat{p}_{s,t}^{(k)} \to p_{s,t}$ a.s. as $k \to \infty$.

IV. Approximating optimal solution

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. (2), 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 $q_s(H_s, \delta(H_s, v_s)) = 1/|\mathcal{R}_{H_s}|$.
- `high-prob-sampling` forms the Markov chain with $q_s(H_s, \delta(H_s, v_s)) = w(\delta(H_s, v_s), H_s) / \sum_{u \in \mathcal{R}_{H_s}} w(\delta(H_s, u), H_s)$.

  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 VI, 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 n, randomly sample a node u from $V(H_n)$ (instead of sampling from $\mathcal{R}_{H_n}$). 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_n and if it is zero, we randomly sample again from $V(H_n)$.

We propose the following algorithms for finding *the partial order based on the estimates of p_{uv}*.

`sort-by-`p_{uv}`-sum` algorithm. We first construct a new complete graph with the node same as that of H_n and edge weights as p_{uv}. Let us now define $p_u := \sum_{v \in V(H_n)} p_{uv}$ 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.

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 placed into the output partial order. Note that if $\tau = 0.5$, we get a total order.

V. Duplication-Divergence Model

We consider Pastor-Satorras et al. definition of the DD-model [14]. 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 evolves from the G_k as follows: first, a new vertex v is added to G_k. Then the following steps are carried out: (i) *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. (ii) *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 $G_n \sim$ `DD-model`(n, p, r, G_{n_0}).

The posterior probability w needed for the importance sampling can be computed as follows. For a node $v_n \in V(H_n)$, the probability of having the node u as its parent in the above model is defined as

$$w(\delta(H_n, v_n), u, H_n) = \frac{1}{n-1} p^{|\mathcal{N}(v_n) \cap \mathcal{N}(u)|} (1-p)^{|\mathcal{N}(u) \setminus \mathcal{N}(v_n)|}$$

$$\times \left(\frac{r}{n-1}\right)^{|\mathcal{N}(v_n)\setminus\mathcal{N}(u)|} \left(1-\frac{r}{n-1}\right)^{(n-1)-|\mathcal{N}(v_n)\cup\mathcal{N}(u)|}. \quad (3)$$

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

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

A. Greedy algorithms for the DD-model

To form a comparison with algorithms proposed in Section IV, we propose the following greedy algorithms for temporal ordered clustering (partial node arrival orders).

`sort-by-degree`. The nodes are sorted by the degree and arranged into clusters $\{C_i\}_{i\geq 1}$. 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 $|\mathcal{N}(v)\setminus\mathcal{N}(u)| \leq r$ 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 $\{u : \nexists v |\mathcal{N}(v)\setminus\mathcal{N}(u)| \leq r\}$ (as mentioned before, it is sufficient to check $\mathcal{N}(v) \subset \mathcal{N}(u)$ 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.

VI. Experiments

In this section, we evaluate our methods on synthetic data (real-data results are available in [4]). We made publicly available all the code and data of this project at [15]. 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 [4] for details), 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 III already captures the MLE solutions and outputs it if it achieves high precision.

In Figure 1 we provide the linear programming (LP) *optimal curve* and approximations to it by our sampling methods. It shows the convergence of the approximated curve obtained through different sampling methods to the exact LP optimal curve. The LP optimal curve (with red color) is given by the optimal precision values computed from the relaxed LP formulation and exact p_{uv} values for various ε – minimum value of density (see Section III-A). Here, σ_{tries} denote the number of Markov chain sample paths used for estimating p_{uv}. We observe that while `local-unif-sampling` method requires only 100 samples for convergence, `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. We consider a small size example here since the total possible number of orderings that is needed for the exact calculation of p_{uv} is $n!$.

In Figure 2, we present results of the p_{uv}-based 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 (see [4], 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 p_{uv}`-threshold` algorithm offer a trade-off between higher precision and higher density – large bin size or the high threshold leads to a decrease in the density, but increase in the precision.

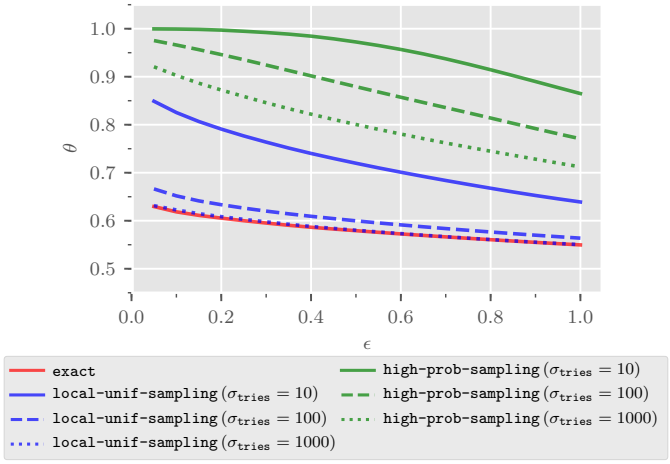

Fig. 1: Comparison of p_{uv} estimation algorithms with LP solution: $G_n \sim$ `DD-model`$(13, 0.3, 1.0, G_{n_0})$, averaged over 100 graphs. G_{n_0} is Erdős-Renyi graph with $n_0 = 4$ & $p_0 = 0.6$.

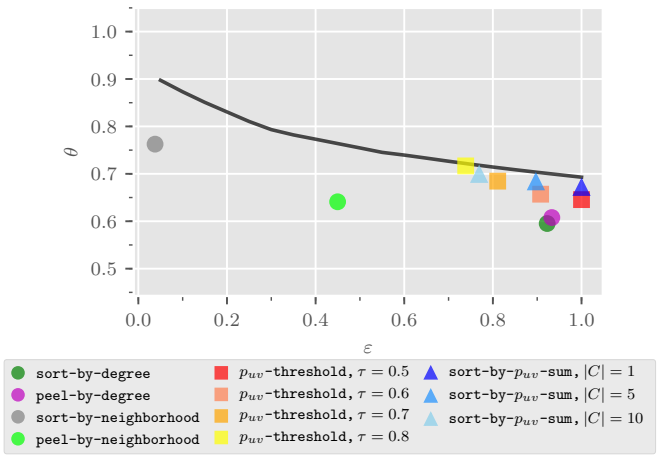

Fig. 2: Comparison of greedy and p_{uv}-based algorithms: $G_n \sim$ `DD-model`$(50, 0.6, 1.0, G_{n_0})$ averaged over 100 graphs. p_{uv}-based algorithms use $\sigma_{\text{tries}} = 100{,}000$. G_{n_0} is generated from Erdős-Renyi model with $n_0 = 10$ and $p_0 = 0.6$. The theoretical curve is estimated via `local-unif-sampling`.

Acknowledgement

This work was supported by NSF Center for Science of Information (CSoI) Grant CCF-0939370, and in addition by NSF Grant CCF-1524312, and National Science Center Grant UMO-2016/21/B/ST6/03146.

References

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

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

[3] M. Srivastava, O. Simakov, J. Chapman, B. Fahey, M. E. Gauthier, T. Mitros, G. S. 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.

[4] K. Turowski, J. K. Sreedharan, and W. Szpankowski, "Temporal ordered clustering in dynamic networks," Extended version. Available at https://www.cs.purdue.edu/homes/jithinks/files/publications/temporal_ordered_clustering_extended.pdf.

[5] 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.

[6] 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.

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

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] Code and data of this submission, available at https://github.com/krzysztof-turowski/duplication-divergence.