Inferring the node arrival sequence from a snapshot of a dynamic network is an important problem, with applications ranging from identifying sources of contagion to flow of capital in financial transaction networks. Variants of this problem have received significant recent research attention, including results on infeasibility of solution for prior formulations. We present an alternate formulation that admits probabilistic solutions for broad classes of dynamic network models. Instantiating our framework for a preferential attachment model, we formulate suitable optimization problems for conventional measures of precision and recall. Noting the computational hardness of these optimization problems, we present efficient algorithms for partial recovery of node arrival orders, and derive rigorous performance bounds on the precision and recall of our methods. We validate our results through experiments on both synthetic and real networks to show that their performance is robust to model changes and gives nontrivial results in practice.

1 Introduction

The problem of inferring the arrival order of nodes in a dynamic network, from its current state, is of considerable significance. In a network of financial transactions, the arrival order tracks the flow of capital. In mapping spread of infectious diseases, arrival order allows one to identify early patients, yielding clues to genetic origin and mechanisms of transmission. In social networks, one can map the spread of information, and use it to design personalized channels. In networks of biochemical reactions (e.g., protein interaction networks), one can identify early biomolecules (e.g., proteins in prokaryotic species), which are known to be preferentially implicated in cancers and other diseases [14].

Owing to this significance, and the availability of large datasets in diverse application domains, there has been considerable recent research interest in variants of this problem, which we discuss in a prior work section below.

Despite the apparent simplicity of the informal problem statement, the node arrival inference problem is highly complex, both from analysis and methodological points of view. This complexity stems, in part, from the symmetries inherent in graphs and the models that generate them, which admit multiple (partial) node orderings with equal likelihood. Characterizing these symmetries and suitably reformulating the problem to establish limits on inference is at the core of the analysis framework. Efficient methods that achieve the prescribed limits of inference characterize the algorithmic challenges associated with the problem.

Dynamic graph models underlying a large class of applications primarily consider node arrivals along with corresponding edge insertions (without corresponding node departures). Starting from a snapshot of a dynamic network, the problem of inferring the arrival order inverts the arrival process. Consequently, a specific inference technique is intimately coupled with a corresponding model of...
the arrival process. In our work, we posit that networks evolve according to some stochastic model, which provides the basis for node order inference, with associated probabilistic guarantees. To this end, we present a general framework for inferring node orders, and a specific instantiation of our framework in the context of a preferential attachment model. We note that our framework is general, capable of admitting a large class of dynamic graph models. Our specific instantiation of preferential attachment graphs is directly applicable to a large number of applications, and has been extensively studied in literature. We will primarily consider the following form of the model:

**Barabási-Albert model** Let \( PA(n, m) \) denote the following version of the Barabási-Albert model \([1]\): the parameter \( n \) represents the number of nodes in the network and \( m \) is the number of connections a new node makes to existing nodes when it is added to the network. To form \( G_t \) for \( t = 1 \) (\( G_1 \) being the graph at time \( t \)), a single vertex (called 1) is created with \( m \) self loops, and at time \( t > 1 \), vertex \( t \) joins the network and makes \( m \) connections to existing nodes in the graph \( G_{t-1} \) with probability proportional to their current degrees as follows: each connection choice is conditionally independent of the others and satisfies \( \Pr[t \text{ connects to } k | G_{t-1}] = \frac{\deg_{t-1}(k)}{2m(t-1)} \), where \( \deg_{t-1}(k) \) is the degree of node \( k \) at time \( t-1 \). Then we define \( G = G_n \). Most of this paper will be based on \( PA(n, m) \) unless otherwise stated explicitly.

**Cooper-Frieze model** We also study a generalized version of the preferential attachment model to demonstrate the robustness of algorithms proposed for \( PA(n, m) \). This is the random graph model proposed in \([6]\) and is an extension of the web graph model \([7]\) (we leave the details of the model to the cited works). Here \( m \) is drawn from a probability distribution. At each time instant, the model allows either addition of a new node or addition of edges between exiting nodes. In both cases, endpoints of new edges can be selected either preferentially or uniformly among existing nodes, according to some parameters.

**Our Contributions**

Magner et al. \([10]\) present a formulation of the problem of recovering the temporal order of nodes in a dynamic graph. Focusing on deriving fundamental limits on inference of temporal order, they show that any estimator, with high probability, makes a large number of mistakes (to be made precise later) – owing to inherent symmetries in the network model. In view of this negative result, we present the following key results:

- We present an alternate formulation that aims to infer a partial order on nodes in terms of their arrival order. We present detailed analytical results on the limits achievable in this new formulation in terms of precision and recall and show that a maximum likelihood estimation of a total order of arrivals leads to an exponentially large set of equiprobable solutions.

- We present an integer programming formulation of optimal solution to our problem. For reasons of computational efficiency, we consider more intuitive and efficient algorithms: the first is optimal in the sense that yields an estimator with perfect precision (see Theorem \([1]\)). That is, given a relabeled preferential attachment graph, it infers all vertex order relations that hold with probability 1. We rigorously analyze the number of such relations and find them to be asymptotically small (Theorem \([3]\)). This motivates our exploration of other algorithms, which sacrifice some precision in order to achieve higher recall. Simulations are given to display the robustness of these algorithms (see Section \([4]\)).

**Prior Related Work** A number of prior results in the area focus on the problem of finding the oldest node in a graph \([3,8]\). Bubeck et al. \([3]\) consider uniform attachment and preferential attachment trees, and find a set that contains the root node with a given error probability. They prove that the set size is independent of the number of nodes. However, their technique cannot be readily extended to the case \( m > 1 \). Our proposed algorithms target a more general problem, and can be used to find such a set for the oldest node. For preferential attachment graphs, Frieze et al. \([8]\) study the problem of identifying the oldest node. Their aim was design an algorithm to identify a small set of nodes (of size polylogarithmic in \( n \)) that contains the root node. A related problem of detecting information sources in epidemic networks has been studied by Shah et al. \([13]\) for the Susceptible-Infected model and Zhu et al. \([15]\) for the Susceptible-Infected-Recovered model. The more general problem of inferring node arrival orders has been shown to be particularly challenging by Magner et al. \([10]\), who derive fundamental bounds for exact and approximate recovery of vertex ordering in preferential

---

1We will drop the subscript \( t-1 \) in \( \deg_{t-1}(k) \) if it is implied from the context.
attachment graphs. Recognizing this complexity, we focus on analysis and algorithmic approaches for partial ordering on vertices. Mahantesh et al. [1] empirically studied inferring arrival order via a binning method; however we provide a rigorous formulation and analysis, a simpler solution without having to generate samples from estimated random graph model, and more precise performance information about our algorithms. To the best of our knowledge, our paper presents the first feasible and rigorous approach to the problem of inferring partial orders in preferential attachment graphs.

2 Problem Formulation

Let $G$ be a graph drawn from the model $\mathcal{PA}(n, m)$. Graph $G$ has a set of nodes $[n] = \{1, \ldots, n\}$, where nodes are listed in arrival order (label 1 is assigned to the oldest node and label $n$ denotes the youngest node). Next, the nodes are subjected to a permutation $\pi$ drawn uniformly at random from the symmetric group $S_n$, and we are given the graph $G' := \pi(G)$; that is, the nodes of $G$ are randomly relabeled. This paper aims to develop methods for inferring arrival order in $G$ after observing $G'$, i.e., to find $\pi^{-1}$. The permutation $\pi^{-1}$ can be considered the true arrival order of the nodes of the given graph; that is, $\pi^{-1}(u)$ for a node $u$ in $G'$ gives the arrival time of that node in the original graph process.

In the recent work of Magner et al. [10], a solution to this problem takes the form of an estimator function $\phi : \mathcal{S}_n \to S_n$, where $S_n$ is the symmetric group on $n$ letters and $\mathcal{S}_n$ is the set of graphs on $n$ vertices. Magner et al. evaluated estimators based on the expected Kendall $\tau$ distance between $\phi(\pi(G))$ and $\pi^{-1}$. In particular, they showed that any such estimator makes, with high probability, $\Theta(n^2)$ inversion errors (note that the maximum possible number of such errors is $\binom{n}{2} = \Theta(n^2)$), owing to the existence of a superexponentially large number of equiprobable graphs with the same structure as $\pi(G)$.

In view of this negative result, we relax the problem: instead of requiring that an estimator infer a total order (i.e., a permutation), we study estimators that output partial orders on the set of vertices (for a partial order $\sigma$, a relation $u <_\sigma v$ indicates that $\pi^{-1}(u) < \pi^{-1}(v)$). The advantage of allowing partial orders is that the ability of an estimator to refrain from guessing the relative order of two vertices allows for the possibility of greater precision.

In evaluating a partial order $\sigma$ as a solution to our problem, we say that an ordered pair of vertices $(u, w)$ in $\pi(G)$ satisfying $\pi^{-1}(u) < \pi^{-1}(w)$ forms a correct pair if $u <_\sigma w$. A pair of vertices is unguessed with respect to $\sigma$ if they are unrelated in $\sigma$. Note that, given a partial order $\sigma$, we can always algorithmically find a total order consistent with $\sigma$ (i.e., a linear extension of $\sigma$) and containing at least as many correct pairs (but generally with more incorrect pairs as well).

A partial order may be described by a directed, acyclic graph (DAG) on the set of vertices. We find it useful to describe a partial order by a binning, which is simply a more compact representation of a DAG: vertices within a given bin are unrelated in the partial order, while an edge from a bin $B_1$ to another one, say, $B_2$, indicates that $v_1 <_\sigma v_2$ for every $v_1 \in B_1$ and $v_2 \in B_2$.

Why is maximum likelihood estimation not a good approach? In the case of total order estimators, a natural way to approach the problem is to frame it in terms of maximum likelihood estimation (MLE) as follows: $C_{\text{ML}}(H) = \arg\max_{\pi \in S_n} P\{G = \pi^{-1}(H) | \pi(G) = H\}$. The following proposition gives a characterization of the optimal solution set $C_{\text{ML}}$. Its proof can be found in the supplementary materials.

**Proposition 1** (MLE of total orders). The maximum likelihood estimation solution set $C_{\text{ML}} = C_{\text{ML}}(\pi(G))$ satisfies $|C_{\text{ML}}| = e^{n \log n - O(n \log \log n)}$ with high probability.

Thus, $C_{\text{ML}}$ gives a large number of equiprobable solutions and the maximum likelihood formulation is not the appropriate approach for the problem.

**Measures for evaluating partial order estimators** In order to evaluate estimators that output partial orders, we define two measures that capture important aspects of their performance:

**Measure of recall:** This measure gives the expected fraction of correct pairs (out of the total number, $\binom{n}{2}$) given by the algorithm that outputs the partial order $\sigma$. It can be interpreted as a modified version of the Kendall $\tau$ metric for partial orders. It is defined as

$$\rho(\sigma) = \mathbb{E} \left[ \frac{1}{\binom{n}{2}} |\{(u, v) \in [n] : u <_\sigma v, \pi^{-1}(u) < \pi^{-1}(v)\}| \right] ,$$

(1)
where \( u <_\sigma v \) means that \( u \) is less than \( v \) in the partial order \( \sigma \).

**Measure of precision:** This measures the expected fraction of correct pairs out of all pairs that are guessed by the partial order. It is given by

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

(2)

where \( K(\sigma) \) denotes the number of pairs \( \{ u, v \} \) that are ordered by \( \sigma \); \( K(\sigma) = | \{ u, v \} : u <_\sigma v \} | \).

The precision of an estimator measures the fraction of all guessed pairs that are correct.

We seek estimators that are optimal in certain respects. Optimality can be defined in several ways: we can optimize the recall (ignoring the precision) to find the largest \( \beta \geq 1/2 \) for which there exists an estimator \( \phi \) with \( \rho(\phi) \sim \beta \). Alternatively, we could choose to optimize the precision under an additional constraint that the estimator must make a minimum number \( W \) of guesses. The latter is appealing because it allows the user of such an estimator to dictate the relative importance of precision versus recall by varying \( W \). In the following section, we consider variations on these optimization problems, with additional restrictions to cope with issues of computational complexity.

### 3 Main Results

We formalize the partial node arrival order inference problem as an optimization problem for maximizing recall and precision (we present results for recall, but the analysis extends to optimizing precision as well). While the underlying integer programming problem does not yet yield tractable algorithms, the formulation yields insights into how order information is encoded in graph structure.

This information is used to derive approximation techniques, which we analyze rigorously. All proofs are relegated to the appendix of the full version, provided in the supplementary materials.

#### 3.1 Optimization for Recall

We assume that our partial orders induce \( n \) (possibly empty) bins, labeled \( B_1, ..., B_n \), arranged in linear order of time; In a partial order \( \sigma \) of this form, we have \( v <_\sigma w \) whenever \( v \in B_i \) and \( w \in B_j \) for \( i < j \). If \( v \) and \( w \) are in the same bin, then they are unrelated by \( \sigma \). Thus, the estimators that we consider will attempt to optimally place vertices into each of the \( n \) bins (note that some bins may be empty, and thus the representation of a given partial order may not be unique).

We can express recall by conditioning on the value of \( \pi(G) \), noting that an estimator is a deterministic function of \( \pi(G) \):

\[
\rho(\phi) = \frac{1}{\binom{n}{2}} \sum_H \text{Pr}[\pi(G) = H] \sum_{1 \leq i < j \leq n} \mathbb{E} \left[ | \{ v \in B_i, w \in B_j : \pi^{-1}(v) < \pi^{-1}(w) \} | \pi(G) = H \right].
\]

(3)

We emphasize that the role of \( \phi \) in the above formula is to determine whether or not a given vertex is in a given bin.

This shows that to maximize the recall, we must choose an estimator \( \phi \) that, on each graph \( H \), allocates vertices to bins so as to maximize the expression

\[
\sum_{1 \leq i < j \leq n} \mathbb{E}[| \{ v \in B_i, w \in B_j : \pi^{-1}(v) < \pi^{-1}(w) \} | \pi(G) = H].
\]

(4)

**Integer programming formulation for maximizing recall** We formulate a binary quadratic program as follows: to each vertex \( v \), associate a vector \( x_v = (x_{v,1}, ..., x_{v,n}) \) taking values in \{0, 1\}^n.

The vector \( x_v \) is the indicator vector for membership of \( v \) in the bins. Then, the optimal estimator maximizes:

\[
\sum_{v \in [n]} \sum_{1 \leq i < j \leq n} x_{v,i} x_{w,j} \text{Pr}[\pi^{-1}(v) < \pi^{-1}(w) | \pi(G) = H],
\]

subject to the following constraints: \( \sum_{j=1}^n x_{v,j} = 1 \ \forall v \in [n] \) and \( x_{v,j} \in \mathbb{Z}_{\geq 0} \ \forall v \in [n], j \in [n] \).

In other words, each vertex is assigned to exactly one bin, and the bin membership vector for each vertex takes non-negative integer values.
This is rather explicit, except for the coefficient \( \Pr[\pi^{-1}(v) < \pi^{-1}(w)|\pi(G) = H] \). The next lemma gives a combinatorial formula for this probability. First, we recall some definitions. Associated with permutations of the random graph model, we define the following terms: 1. Set of feasible permutations of a graph \( H \): the subset \( \Gamma(H) \subseteq S_n \), which consists of permutations \( \sigma \), such that \( \sigma(H) \) has positive probability under the distribution \( \mathcal{PA}(n,m) \). 2. Set of admissible graphs of \( H \): \( \text{Adm}(H) = \{ \sigma(H) : \sigma \in \Gamma(H) \} \). These are the graphs obtained by applying elements of \( \Gamma(H) \) to \( H \).

**Lemma 1** (Expression for coefficients of the quadratic integer program). We have, for all \( v, w \in [n] \) and graphs \( H \),

\[
\Pr[\pi^{-1}(v) < \pi^{-1}(w)|\pi(G) = H] = \frac{\{ \sigma : \sigma^{-1} \in \Gamma(H), \sigma^{-1}(v) < \sigma^{-1}(w) \}}{\left| \Gamma(H) \right|}.
\]

We thus end up with an integer program written in terms of coefficients that are (in principle) explicitly computable, and the expression in Lemma 1 can be used to guide intuition in the design of more efficient heuristic methods.

Several technical challenges remain in the implementation of the above approach: the general problem of integer programming is NP-hard, and exact calculation of the coefficients of the above program reduces to an instance of another hard problem: namely, counting linear extensions of a partial order (the general version of which is known to be \#P-complete \[^2\]). These challenges may be surmountable, but in the end may result in a slow polynomial-time algorithm. In the rest of the paper, we formulate simpler, efficient algorithms and rigorously analyze their performance.

### 3.2 Practical solutions

We start our discussion with the following definition:

**Definition 1** (DAG of \( G \)). For \( G \) distributed according to the preferential attachment distribution (for any \( m \)), we define \( \text{DAG}(G) \) to be the directed acyclic graph defined on the same vertex set as \( G \), such that there is an edge from \( u \) to \( v < u \) if and only if there is an edge between \( u \) and \( v \) in \( G \). This is just \( G \) with the directions of edges remembered (see Figure \[^1\]).

We next describe the PEELING technique which recovers \( \text{DAG}(G) \) from \( G \) and, as a side-effect, produces a natural binning scheme. The algorithm starts by identifying the lowest-degree nodes (in our model, the nodes of degree exactly \( m \)) and places them in the youngest bin. Then, it removes all of these nodes and their edges from the graph. The process proceeds recursively, removing the lowest-degree nodes and placing them in the next-youngest bin, until there are no more nodes to remove from the graph. To construct \( \text{DAG}(G) \) during this process, we simply note that all of the edges of a given degree-\( m \) node in a given step of the peeling process must be to older nodes; hence, their orientations can be recovered. See Lemma \[^2\] An example graph and its DAG with the indication of bins are shown in Figure \[^1\].

**Algorithm 1** Peeling Technique

```plaintext
1: procedure PEELING(G)
2:   t ← 1
3:   while \( \text{NoNodes}(G) > 0 \) do
4:     MinDeg ← \text{FindMinDegree}(G) \hspace{1cm} \triangleright \text{Finds minimum degree}
5:     \( B_t \leftarrow \{ u : \deg(u) = \text{MinDeg} \} \) \hspace{1cm} \triangleright \text{Nodes with minimum degree}
6:     \( G \leftarrow \text{RemoveNodes}(G, B_t) \) \hspace{1cm} \triangleright \text{Removes minimum degree nodes from G}
7:     t ← t + 1
8: end while
9: return \( \{ B_t \} \) \hspace{1cm} \triangleright \text{Returns set of bins}
10: end procedure
```

**Lemma 2** (Reconstruction of \( \text{DAG}(G) \)). The PEELING algorithm exactly recovers \( \text{DAG}(G) \).

\( \text{DAG}(G) \) conveniently encodes the set of all vertex pairs whose true order relationship can be inferred with complete certainty. To make this precise, for a graph \( H \) or its DAG, we define a vertex pair order

\[^2\] In reference to the DAG of a \( \mathcal{PA} \) graph, we ignore self-loops, without loss of generality.
Definition 2 (Perfect vertex pair order event). A vertex pair order event \((u, v)\) is perfect for a graph \(H\) if, for all \(\sigma \in \Gamma(H)\), we have \(\sigma(u) < \sigma(v)\). Equivalently, recalling that \(\pi\) was the uniformly random permutation used to relabel the vertices of \(G\), \(\Pr[\pi^{-1}(u) < \pi^{-1}(v) | \pi(G) = H] = 1\).

The following result formalizes our intuition that \(\text{DAG}(G)\) captures all probability-1 information about \(G\):

Theorem 1 (\(\text{DAG}(G)\) captures perfect vertex pair information). For a DAG \(H\) on the vertex set \([n]\) satisfying \(\Gamma(H) \neq \emptyset\), and for any \(u, v \in [n]\), \((u, v)\) is perfect for \(H\) if and only if there exists a directed path in \(H\) from \(v\) to \(u\) (denoted by \(v \Rightarrow u\)).

Given this interpretation of \(\text{DAG}(G)\), it is natural to ask for the typical number of perfect pairs. This will give the recall of the described estimator.

3.3 Analysis of the number of perfect pairs

The following theorem gives results about the number of perfect pairs. The main message here is that the number of perfect pairs is in general asymptotically negligible compared to the number of correct pairs (which is of the order \(\Theta(n^2)\)).

Theorem 2 (Typical number of perfect pairs). With high probability, for arbitrary fixed \(m \geq 1\), the number of perfect pairs associated with \(G\) is \(\Omega(n \log n)\) and \(o(n^2)\) (uniformly in \(m\)). In the special case of \(m = 1\), we have the matching upper bound of \(O(n \log n)\).

The above theorem is validated by simulation as shown in Figure 2.

Further scrutiny of the numerical evidence leads to a conjecture about the more precise behavior of the number of perfect pairs as a function of \(m\): for \(m > 1\), the number of perfect pairs is \(O(n^{1+\delta(m)})\), for some function \(1 > \delta(m) > 0\) (see again Figure 2).

\(^3\) All simulation results take empirical expected values with respect to several samples from the graph distribution in question.
This conjecture arises from the intuition from the (rigorous) proof of Theorem 2 as follows: denote by $X(u)$ the number of descendants of any vertex $u$ in DAG($G$), and by $X_t$ the number of perfect pairs in the graph up to time $t$. If we can upper bound the number of descendants of any given vertex at time $t$ by $O(t^c)$ (see Figure 3), for some constant $c = c(m) < 1$ dependent only on $m$, then we have $E[X_t] \leq E[X_{t-1}] + m + O(t^c) = E[X_{t-1}] + O(t^c)$. This upper bound arises as follows: the pairs that are perfect at time $t - 1$ remain perfect at time $t$, yielding the first term of the upper bound. The second and third terms arise from the fact that vertex $t$ connects to at most $m$ other vertices (resulting in $m$ additional perfect pairs), and each perfect pair $(v, w)$ involving two descendants of a node chosen by vertex $t$ yields another perfect pair $(v, t)$.

Iterating this recurrence and using Euler-Maclaurin summation, we find that $E[X_t] = O(t^{1+c(m)})$, as desired. We give empirical evidence for the upper bound on $X(t)$ (and hence for the improved upper bound on the number of perfect pairs): in Figure 3 that the maximum number of descendants of any node at a given time $t$ is $O(t^{c(m)})$.

Figure 3: Normalized average and maximum number of descendants of degree-$m$ nodes in DAG($G$).

The main consequence of Theorem 2 is that, for any $m$, an estimator that only outputs perfect pairs has precision exactly 1 and recall only $o(1)$. The poor recall of this method motivates further exploration of the binning produced by the PEELING procedure. The goal is to sacrifice as little precision as possible in order to achieve asymptotically positive recall.

In the case of $m = 1$, $G$ is a tree, and we are able to precisely characterize certain parameters of the PEELING procedure, owing to the large amount of literature surrounding the preferential attachment distribution for this case (the structure is identically distributed to a random plane-oriented recursive tree): for instance, the asymptotic size of the $j$th youngest bin, for each fixed $j$, may be precisely determined by writing it as an additive parameter and using results from Wagner et al. [12]. Theoretical results for $m > 1$ are much more elusive, except that we can show that the peeling process recovers all perfect pairs, in addition to many imperfect ones. Moreover, we can derive significant insight by studying the PEELING method empirically. In what follows, $R_{\text{peel}}$ denotes the binning produced by the PEELING algorithm. Table 1 (and Figure 4) shows simulation results for samples from the Barabási-Albert model.

The performance improves greatly with even small increases in $m$. This can be explained intuitively as follows: for small $m$, as the graph evolves, each new vertex is likely to connect to high-degree nodes, which are already in older bins in DAG($G$). Thus, bins tend to be large, resulting in low precision and recall. When we increase $m$, each new node has a higher probability of choosing a degree-$m$ node (say, $v$), which results in shifting the bin indices of all descendants of $v$. Thus, larger $m$ leads to a DAG with a more equitable distribution of vertices across bins, which yields higher precision and recall.

Figure 4: Correct pairs by PEELING.

<table>
<thead>
<tr>
<th>$m$</th>
<th>$\theta(R_{\text{peel}})$</th>
<th>$\rho(R_{\text{peel}})$</th>
<th>Bins in $R_{\text{peel}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.878</td>
<td>0.758</td>
<td>39.86</td>
</tr>
<tr>
<td>10</td>
<td>0.917</td>
<td>0.858</td>
<td>68.77</td>
</tr>
<tr>
<td>25</td>
<td>0.958</td>
<td>0.936</td>
<td>140.37</td>
</tr>
<tr>
<td>50</td>
<td>0.977</td>
<td>0.967</td>
<td>237.97</td>
</tr>
</tbody>
</table>

Table 1: Comparison for $\mathcal{PA}(n = 5000, m)$
Table 2 demonstrates the robustness of our Peeling algorithm for various generalizations of preferential attachment model including variable $m$ (denoted by $M$ and $\sim \text{unif}$ denote uniform distribution) with uniform attachment (denoted by $\text{UA}$) and the very general Cooper-Frieze model \cite{6,7}.

<table>
<thead>
<tr>
<th>Technique</th>
<th>$\theta(R_{peel})$</th>
<th>$\rho(R_{peel})$</th>
<th>Bins in $R_{peel}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{PA}(n, m = 25)$</td>
<td>0.958</td>
<td>0.936</td>
<td>140.37</td>
</tr>
<tr>
<td>$\mathcal{PA}(n, M \sim \text{unif}{5, 50})$</td>
<td>0.691</td>
<td>0.683</td>
<td>491.81</td>
</tr>
<tr>
<td>$\mathcal{UA}(n, m = 25)$</td>
<td>0.977</td>
<td>0.967</td>
<td>238.23</td>
</tr>
<tr>
<td>$\mathcal{UA}(n, M \sim \text{unif}{5, 50})$</td>
<td>0.827</td>
<td>0.823</td>
<td>707.01</td>
</tr>
<tr>
<td>Cooper-Frieze (Web graph) model\cite{6}</td>
<td>0.828</td>
<td>0.822</td>
<td>619.25</td>
</tr>
</tbody>
</table>

| | \footnotesize{Table 2: A general comparison: $n = 5000$} |
| | These results suggest that the proposed DAG-based methods can simultaneously achieve high precision and recall. |

4 Experimental Results

We evaluate our methods on various real-world networks. Here, $R_{peel}$ again denotes the Peeling estimator, and $R_{peel}$ denotes the estimator that outputs all perfect pairs. All results are in Table 3.

The first is a citation network: the largest connected component of ArXiv High Energy Physics (Theory) directed network released by KDD cup 2003 ($|V| = 7464$, $|E| = 116,268$) \cite{5}. The original ranking of the network is not a full ranking and has 1457 bins. The next dataset is the Simple English Wikipedia dynamic network ($|V| = 100,312$, $|E| = 1,627,472$) \cite{5}. This network shows the evolution of hyperlinks between articles of the Simple English Wikipedia. Nodes in this network represent articles and an edge indicates that a hyperlink was added or removed. We also study a large collaboration graph, DBLP which is an undirected network of authors of scientific papers in DBLP computer science bibliography ($|V| = 1,137,114$, $|E| = 5,018,065$) \cite{5}. An edge between two authors represents a common publication. Finally, we present results for an interaction network for human proteins. This network, from BioGRID\cite{https://thebiogrid.org} is directed ($|V| = 14,867$, $|E| = 126,593$). The original ranking is taken as the approximate ranking provided by ProteinHistorian website\cite{https://lighthouse.ucsf.edu/ProteinHistorian/} which uses the methods proposed in \cite{4}, and has only 16 bins. There are 1054 nodes without age information in ProteinHistorian and they are removed from the network. The ranking provided in \cite{4} is only an approximation, and not the ground truth.

| | \footnotesize{Table 3: Results for citation, Simple English Wikipedia, and protein interaction networks} |
| | For all real networks tested, with the exception of the protein interaction network, our algorithms yield nontrivial precision and recall results, consistent with our theoretical predictions. In the case of protein interaction networks, our definitions of precision and recall are overly pessimistic for a few reasons, the first of which is that the “ground truth” that we use is a very sparse binning. A pair whose order is guessed by our algorithm and unguessed in this ground truth is counted as an error. Thus, we extend the ground truth ordering to a total order by randomly guessing the order of unguessed pairs, resulting in the numbers given. Moreover, importantly, the protein interaction network is much noisier than the others that we analyzed: while we have reasonable confidence that all edges are accounted for in the citation and Wikipedia networks, there is no such guarantee that this holds for the protein interaction network, as the presence or absence of each edge is costly to determine. Finally, in general, the more preferential attachment plays a role in the formation of a real network, the closer the performance will be to our theoretical and synthetic results. In all of the networks that we analyzed, the degree distribution had a power law exponent strictly between 1 and 3.5 \cite{5}, hinting at the existence of a preferential attachment process mixed with edge insertions among existing nodes and uniform attachment \cite{6}. |

\footnotesize{\textsuperscript{4}The parameters used and the details for generating the model are explained in Supplementary material}

\footnotesize{\textsuperscript{5}https://thebiogrid.org}

\footnotesize{\textsuperscript{6}https://lighthouse.ucsf.edu/ProteinHistorian/}
References


APPENDIX

5 Proofs

5.1 Proof of Proposition \[1\]

First, by definition of $\Gamma(H)$, we must have that $C_{ML}(H) \subseteq \Gamma(H)$. We will show, in fact, that the likelihoods given to all elements of $\Gamma(H)$ are equal. This will then imply that $C_{ML}(H) = \Gamma(H)$.

Next, from a result of \[9\], we have that with high probability $|\Gamma(\pi(G))| = e^n \frac{\log n}{O(n \log \log n)}$, which will complete the proof.

So it is sufficient to show that, for each $\sigma \in \Gamma(H)$,

$$\Pr[G = \sigma(H)|\pi(G) = H]$$

depends only on the structure $S(H)$. To do this, note that by definition of $\text{Adm}(H)$ and $\Gamma(H)$, we must have $\sigma(H) \in \text{Adm}(H)$. So it is enough to show that for any two positive-probability isomorphic graphs $G_1$ and $G_2$, we have $\Pr[G = G_1] = \Pr[G = G_2]$. This is the content of the following lemma.

**Lemma 3** (All positive-probability isomorphic graphs are equiprobable). Consider two isomorphic graphs $G_1, G_2 = \phi(G_1)$. If $\Pr[G = G_1] > 0$, then

$$\Pr[G = G_1] = \Pr[G = G_2].$$

(7)

**Proof.** First, we will show that for a positive-probability graph $g$, $\Pr[G = g]$ depends only on the unlabeled DAG of $g$ (see Definition 1 for the definition of the labeled version). Then, it is implicit in the proof of Lemma 2 that the unlabeled DAG of $g$ is uniquely determined by $S(g)$, which will complete the proof.

To show that $\Pr[G = g]$ depends only on the unlabeled DAG of $g$, we derive an explicit expression as follows: suppose that for a given vertex $v$ of $g$, the number of neighbors of $g$ that chose it is denoted by $N_g(v)$ (we call such neighbors parents). Furthermore, let $d_1(v) \leq \cdots \leq d_{N_g(v)}(v)$ denote the number of edges from each of the parents of $v$ to $v$ (note that all of this information is given by the unlabeled DAG of $g$). Finally, let $K_g(v)$ denote the number of orders in which the parents of $v$ could have arrived in the graph. Note that this is solely a function of the unlabeled DAG of $g$.

$$\Pr[G = g] = \frac{\prod_{d \geq m} \prod_{v : \text{deg}_g(v) = d} K_g(v) \prod_{i=1}^{N_g(v)} \binom{m}{d_i} \prod_{h=0}^{d_i-1} (m + d_1 + \cdots + d_{i-1} + h)}{\prod_{j=1}^{m-1} (2m_j)}$$

(8)

Here, $\text{deg}_g(v)$ denotes the degree of vertex $v$ in the graph $g$. The first product is over all degrees present in the graph $g$, the second is over all vertices $v$ having the given degree, the third is over all parents of $v$, and the fourth gives the contribution of the connection choices from each given parent of $v$. Since all parts of this expression are dependent only on the structure of DAG($g$), the lemma is proven.

This completes the proof of the proposition.

5.2 Proof of Lemma \[1\]

We can express the conditional probability in question as a sum, as follows:

$$\Pr[\pi^{-1}(v) < \pi^{-1}(w)|\pi(G) = H] = \sum_{\sigma : \sigma^{-1} \in \Gamma(H), \sigma^{-1}(v) < \sigma^{-1}(w)} \Pr[\pi = \sigma|\pi(G) = H].$$

Now, recall that $\pi^{-1} \in \Gamma(H)$ under this conditioning, since $\pi(G) = H$ and $G$ is admissible. Moreover, it is uniformly distributed on $\text{Iso}(G, H)$ (the set of isomorphisms from $G$ to $H$), so we have

$$\Pr[\pi = \sigma|\pi(G) = H] = \frac{\Pr[G = \sigma^{-1}(H)|\pi(G) = H]}{|\text{Aut}(H)|} = \frac{1}{|\text{Aut}(H)||\text{Adm}(H)|}.$$  

---

\[1\] For definitions of admissibility, $\Gamma$, and $\text{Adm}$, see the text before Lemma 1.
Taking the sum, this becomes
\[
\frac{|\{\sigma : \sigma^{-1} \in \Gamma(H), \sigma^{-1}(v) < \sigma^{-1}(w)\}|}{|\text{Aut}(H)| |\text{Adm}(H)|}.
\]
Finally, recall that \(|\text{Adm}(H)| = \frac{|\Gamma(H)|}{|\text{Aut}(H)|}\). This gives the desired result.

5.3 Proof of Theorem 1

First, we will show that if \(v \rightsquigarrow u\), then \((u, v)\) is perfect. We will do this by first showing the simpler claim that if there is an edge from \(v\) to \(u\) (denoted by \(v \rightarrow u\)), then \((u, v)\) is perfect. This is sufficient, since then \(v \rightsquigarrow u\) implies that there is a directed path \(v = v_0 \rightarrow v_1 \rightarrow \cdots \rightarrow v_k \rightarrow u = v_{k+1}\) in \(H\), which implies that \((v_i, v_{i+1})\) is perfect for each \(i \in \{0, \ldots, k\}\). Then we observe that perfectness is a transitive relation, which implies the desired result.

So it remains to show the claim assuming that \(v \rightarrow u\). Let \(\sigma \in \Gamma(H)\). This means that \(\sigma(H)\) is a positive-probability DAG under the preferential attachment model. Note also that \(\sigma(v) \rightarrow \sigma(u)\) in \(\sigma(H)\), which implies that we certainly must have \(\sigma(v) > \sigma(u)\) (vertices only choose to connect to older vertices). Since \(\sigma\) was arbitrary, we have that \((u, v)\) is perfect for \(H\), as desired.

Now we show the converse claim: if \((u, v)\) is perfect for \(H\), then \(v \rightsquigarrow u\) in \(H\). We will do this by proving the contrapositive: assume that there is no directed path from \(v\) to \(u\). Then we will construct a permutation \(\sigma\) satisfying

\begin{enumerate}
  \item \(\sigma \in \Gamma(H)\),
  \item \(\sigma(u) > \sigma(v)\).
\end{enumerate}

This is equivalent to producing a feasible schedule of the vertices of \(H\) (i.e., a sequence of distinct vertices \(v_1, v_2, \ldots, v_n\) of \(H\), such that, for each \(j \in [n]\), all \(m\) of the the descendants of \(v_j\) in \(H\) are contained in the set \(\{v_1, \ldots, v_{j-1}\}\)). We will require that \(v_u > v_v\) in the schedule. Such a schedule gives a permutation satisfying the properties above as follows: for each \(j\), \(\sigma(j) = v_j\).

To do this, we start by considering the sub-DAG \(H_v\), consisting of \(v\) and all of its descendants. Now, we set \(v_1\) to be the bottom vertex of \(H\) (which is also the bottom vertex of \(H_v\)). We will add subsequent vertices to our schedule as follows: at each time step \(t > 1\), \([n]\) is partitioned into three parts: \(S_{p,t}\) (the vertices already in the schedule), \(S_{a,t}\) (the active vertices; i.e., those vertices not in \(S_{p,t}\), all of whose children in \(H\) are contained in \(S_{a,t}\)), and \(S_{d,t}\) (the dormant vertices; i.e., those vertices that are not active or already processed). So \(S_{p,1} = \{v_1\}\), \(S_{a,1}\) is the set of neighbors of \(v_1\), and \(S_{d,1}\) consists of the rest of the vertices.

We observe that \(S_{a,t}\) is nonempty unless \(t = n\): otherwise, there are \(< n\) vertices in \(S_{p,t}\) (in fact, precisely \(t\) of them), and the rest are in \(S_{d,t}\). In this case, consider the lowest vertex \(\ell\) in \(S_{d,t}\); \(\ell\) cannot have any children in \(S_{d,t}\), since it is the lowest, so all of its children must be in \(S_{p,t}\). But this means precisely that \(\ell \in S_{a,t}\). Thus, \(S_{a,t}\) must be nonempty.

Note that it is clear that at any time \(t\), we can designate any active vertex as the next one in our schedule; we would then move it to the processed set, potentially resulting in some vertices in \(S_{d,t}\) becoming active.

Now, observe that at time \(t = 1\), some vertex from \(H_v\) must be active (by the same reasoning that established that the active set must be nonempty). In fact, until all vertices of \(H_v\) have been processed, there remains at least one such vertex that is active. We thus choose active vertices \(H_v\) until it is entirely processed (note that we do not process the vertex \(u \notin H_v\) yet, since there is no directed path from \(v\) to \(u\)). Then we process active vertices until a complete schedule has been generated. By construction, \(v\) comes earlier in the schedule than \(u\), which completes the proof.

5.4 Proof of Theorem 2

Upper bound: Let \(X_t\) denote the number of perfect pairs in the graph immediately after time \(t\). We will prove the claimed upper bound by upper bounding \(X_t\) in expectation, then using Markov’s inequality. To bound \(E[X_t]\), we will show that it is sufficient to upper bound \(X(v)\) in expectation for each \(v < t\), where \(X(v)\) denotes the number of descendants of \(v\) in the DAG.
From Proposition 1 of [9], if we take $\ell = \Theta(\log^{4/5} n \log(\log n))$ (so that $\delta = \frac{1}{\log^{1/5} n}$), then we see that all but $O\left(\frac{n}{\log^{1/5} n}\right)$ vertices are contained in the first $\ell$ levels of the DAG (the levels in that paper are equivalent to the bins in our paper, where the youngest bin is the first level). This fact allows us to upper bound $X(u)$ of any given vertex $u$ as follows: the number of its descendants contained in the first $\ell$ levels is at most $O(m^\ell) = o(n)$, and the number of descendants in larger levels is upper bounded by the total number of vertices in those levels $- O(n/\log^{1/5} n)$. The second of these quantities dominates, so that the total number of descendants $X(u)$, for all $u \leq n$ is at most $O(n/\log^{1/5} n)$, with high probability and in expectation.

Now, we translate this to an upper bound on $X_t$ as follows: we have

$$E[X_t] \leq E[X_{t-1}] + m + mO(t/\log^{1/5} t).$$

This upper bound is from the following facts: at time $t$, all perfect vertex pairs from time $t - 1$ are still perfect, contributing the $X_{t-1}$ term. Next, vertex $t$ makes at most $m$ choices, creating at most $m$ new perfect pairs (which explains the second term). Finally, the third term comes from the fact that if $t$ chooses $v$, and $u$ is a descendant of $v$ (so that $(u,v)$ is a perfect pair), then $(u,t)$ is also perfect.

Solving this recurrence, we find that $E[X_t] = o(t^2)$, as desired, and the proof is completed using Markov’s inequality.

In the case where $m = 1$, we have a much better upper bound on $X(v)$, for arbitrary $v$: with high probability, at time $t$, $X(v) = O(\log t)$, as a result of the height of the tree being $O(\log t)$. This gives the improved bound of $E[X_t] = O(t \log t)$.

**Lower bound:** To prove the lower bound of $\Omega(t \log t)$ perfect pairs, we write the total number of perfect pairs in such a graph as the sum, over all vertices $v \leq t$, of the number of descendants $X(v)$:

$$X_t = \sum_{v \leq t} X(v).$$

In the $m = 1$ case (where $G$ is a tree), this implies that $X_t$ is the total path length of the tree. It is known that, with high probability, this parameter is $\Theta(t \log t)$ (since it can be written as an additive parameter, the results of [12] apply).

Now, to prove it for general $m$, we recall that to form a sample $G$ from $\mathcal{PA}(n,m)$, we first produce a sample $\tilde{T}$ from $\mathcal{PA}(mn,1)$, then collapse sequences of $m$ consecutive vertices. The number of perfect pairs in $G$ is at least the same quantity in $\tilde{T}$, divided by $m^2$, since there can be at most $m^2$ edges from one sequence of $m$ consecutive vertices to an older sequence in $\tilde{T}$. Using the result for $m = 1$, we have that with high probability, the number of perfect pairs in $T$ is $\Theta(m t \log(m t)) = \Theta(t \log t)$, which implies that the number of perfect pairs in $G$ is $\Omega(t \log t)$.

5.5 **Proof of Lemma 2**

The peeling algorithm maintains the following invariant at the beginning of each step: every degree-$m$ node connects only to vertices older than itself in the remaining graph. This is clear in the initial step, since a node can only have degree exactly $m$ in the full graph if its neighbors are all at least as old as it is. In subsequent steps (assuming by induction that the invariant holds for all previous ones), if some edge adjacent to a degree-$m$ node $u$ is adjacent to a younger node $v$ in the current graph, then this implies that some node $w$ older than $u$ and adjacent to $u$ in a previous step has already been removed. This, in turn, implies that in that previous step, $w$ had degree $m$ and was connected to $u$, a younger vertex. This yields a contradiction, completing the proof.

5.6 **Cooper-Frieze model**

For simulating Cooper-Frieze model, we use the following parameters:

- $T$, total number of time epochs: $T = 5000$
- At each time instant, with probability $\delta$, a new node is added, and with probability $1 - \delta$, new edges are created among existing nodes. $\delta = 0.75$
• With probability $\gamma$, terminal node of an edge from the new node is selected preferentially from existing nodes, and with probability $1 - \gamma$, terminal node is selected uniformly. $\gamma = 0.5$

• $p$: Distribution for the number of edges when a new node is added. $p = \text{Uniform}\{5, 50\}$

• $q$: Distribution for the number of edges when edges are added between existing nodes. $q = \text{Uniform}\{5, 50\}$

• When adding an edge between existing nodes, with probability $\delta'$, source node of the edge is selected preferentially among the existing nodes and with probability $1 - \delta'$, source node is selected uniformly. $\delta' = 0.5$

• When adding an edge between existing nodes, with probability $\gamma'$, terminal node of the edge is selected preferentially among the existing nodes and with probability $1 - \gamma'$, terminal node is selected uniformly. $\gamma' = 0.5$

• The probabilities $\gamma, \gamma', \delta'$ are invoked each time a new edge is added.