Independent Work Report

Charlie Hou (Advised by Prof. Miklos Racz)

May 27, 2018

1 Abstract

Given that a graph is generated by the Barabasi-Albert model, it is of theoretical and practical interest to recover the order in which nodes arrived to the graph. This problem has been studied by Magner et al. in [1] and [2], where they introduced the peeling and peeling+ algorithms to recover vertices. Those algorithms perform close to the theoretical upper bound on recovery that they found. In this independent work project, we create a family of algorithms that interpolate between the peeling and peeling+ algorithms, which gives us more control over the desired recall and precision and sheds light on the lower bound for recovery. Furthermore, we shed some light on the peeling algorithm both theoretically and experimentally.

2 Introduction

Vertex order recovery is an important statistical inference problem. For example, knowing the arrival order of fake news in a fake news network can help a social network like Facebook or Twitter understand fake news sources. Or, given a network of infected computers, finding an order of the vertex arrivals can give clues on the spread of the virus. In general, any time where understanding the evolution of a network is important, figuring out the order in which the nodes arrived given the last network is of great practical advantage. Because of the practical significance of the problem, there has been much research interest on this problem.

The bulk of this research centers on seed graphs of randomly growing graphs, or the graph that exists at time step 1. Borgs et al. found a polylogarithmic (in the number of queries) algorithm to find the seed in the preferential attachment model [3], Bubeck et al. show that for all algorithms that produce a set of K vertices that contain the first vertex w.p. $1 - \epsilon$, the best K is subpolynomial in ϵ in uniform attachment and polynomial in $1/\epsilon$ in preferential attachment [4]. Finally, Bubeck, Mossel, and Racz [5] show that different seed graphs in the preferential attachment model lead to different graphs in the limit. We expand on the literature in this field by studying the generalization of this problem: inferring a vertex ordering from beginning to end of a random graph, given a snapshot.

We Magner et al. found that the problem cannot be perfectly solved [2]: with high probability, an algorithm that tries to perfectly recover the ordering of the vertices will make mistakes. This is because (loosely speaking) graphs are by their nature very symmetric and have many isomorphisms. So there's many different possibilities of vertex orderings given the last graph. What we can hope for, however, is a partial ordering, where judgement is withheld on certain pairs of vertices.

They introduce three different algorithms: the perfect pair algorithm, which guesses very little but gets all its guesses correct, the peeling algorithm, which makes a lot of guesses and gets a lot correct (to be made precise later), and the peeling+ algorithm, which makes guesses on all of the vertices, and whose precision is a little below that of the peeling algorithm.

While this problem is relevant for many different random graph models, we choose to study it in the context of the Barabasi-Albert model, because it is a good model for applications like social networks and disease networks, which are the primarily the types of applications that vertex order recovery is useful for. Let $PA(n, m)$ be the probability distribution of graphs that are generated by the Barabasi-Albert model with n vertices and m edges. More precisely, given graph $G^{(t)}$, $G^{(t+1)}$ is generated by introducing a new vertex and forming m edges with vertices in the graph. The probability that the new vertex, v, forms an edge with a vertex v' is given as follows:

$$P(v\ forms\ edge\ with\ v') = \frac{deg(v')}{2mt} \tag{1}$$

At $t = 1$, a single vertex simply has m self-loops, to make this model consistent with the model given in [1].

Magner et al. [1] introduces the peeling and peeling+ algorithms. The full algorithm is listed in the paper, so we will not repeat it in detail here. Instead, we will give a quick overview. Let $G^{(t)}$ be the graph that we are performing the algorithm on. At the first step of the algorithm, we take all the vertices that have the lowest degree, remove them from the graph, and put them into one "bin", and we call this bin one. Repeat this process until the graph is empty, and we get a series of bins. Let $G_i^{(t)}$ be the graph $G^{(t)}$ after $i-1$ steps of the peeling algorithm; note $G^{(t)} = G_1^{(t)}$. Let $B_i^{(t)}$ be the bin formed after performing the ith step of the algorithm. We then consider the lower-numbered bins as the bins with the younger vertices and the higher-numbered bins as the bins with the older vertices.

We call "precision" the number of guesses that an algorithm performing vertex recovery gets correct, and "recall" the number of guesses that an algorithm gets correct total. Peeling empirically performs very close to the upper bound of the precision curve for all algorithms (the curve generated by treating precision as a function of the number of guesses the algorithm makes) [1], and provably gets all "perfect pairs" (pairs of vertices that are ordered the same way under all feasible permutations of vertex labels under the distribution generating the graph) correct. In this sense, peeling is a good algorithm and is close to "optimal".

The peeling+ algorithm tiebreaks vertices within a bin by considering what bins a vertex's neighbors are in. If the vertex's neighborhood is has vertices in higher-numbered bins, it is considered to be an older vertex. This algorithm, by tie-breaking, makes guesses about the ordering of almost all the vertices, as bin sizes for the vast majority of bins in this algorithm end up being one. Peeling+ also performs very close to the upper bound of the precision curve [1] empirically.

Our contributions:

-First, we reproduce some of the experimental results from the paper by Magner et al. [1], specifically with the peeling and peeling+ algorithms.

-Second, we introduce a family of algorithms called Interpolated Peeling, which give finer control over the desired precision and recall of the peeling algorithm. We make some preliminary progress on using this algorithm to find a lower bound on the precision curve.

-Third, we make some progress in understanding the peeling algorithm both theoretically and experimentally.

3 Problem Formulation

The following problem formulation is problem is formulated in the Magner paper, and is repeated here in a condensed manner for completeness, with a few changes for convenience for this paper [1]. We consider a graph $G^{(t)}$ that is generated $PA(t,m)$. We consider this graph as fixed. We apply a permutation π to the labels of $G^{(t)}$, which is drawn according to the probabilities given in $PA(t,m)$. In other words, the probability of a specific permutation is the probability that the graph with the permutation as the true labelings could have arisen according to $PA(t,m)$.

Then our job is to find π^{-1} given $\pi(G^{(t)})$. What we want to do then is to find an estimator for π^{-1}. However, π^{-1} is random, so this becomes a statistical inference problem. We want to estimate π^{-1} with ϕ. However, according to [1], actually estimating π^{-1} directly leads to many errors. So we instead use an estimator that finds a partial order σ instead. According to σ, u is older than v if $\sigma(u) > \sigma(v)$. So now, our estimator ϕ tries to find a partial ordering.

There are a few measures that we are interested in to evaluate the quality of an ordering σ. The first would be density, the number of compared pairs over the total number:

$$\delta(\sigma) = \frac{K(\sigma)}{\binom{n}{2}} \tag{2}$$

The second would be precision, where the expectation is taken over the randomness of π.

$$\theta(\sigma) = E[\frac{1}{K(\sigma)}|number\ of\ guesses\ correct\ by\ \sigma|] = E[\frac{\psi(\sigma,\pi)}{K(\sigma)}] \tag{3}$$

And the third is recall:

$$\rho(\sigma) = E[\frac{\psi(\sigma,\pi)}{\binom{n}{2}}] \tag{4}$$

There is obviously a tradeoff between precision and recall, or alternatively between precision and density. In the Magner paper, they mostly looked at the optimal curve between precision and density. In our experiments, we mostly consider the tradeoffs between precision and recall. Both are essentially talking about the same thing.

4 Main Results

4.1 Warmup: Result Reproduction

Naturally, we first tried to reproduce the empirical results from the Magner paper. For the peeling algorithm, they find that the average recall at $m = 5, n = 5000$ to be .758, while we find that our recall tends to be around .775 or so, in figure 1.

Furthermore, the average precision for $m = 5$ that Magner found was on average .954, while we found it to be around .885 in figure 2, and for an average taken over 1000 graphs, we found an average of .878.

In general, we tended to find lower precision values and higher recall values for each value of m compared to the paper.

Next, we wanted to try to reproduce the results given by peeling+ in the paper. They do not give a table in the paper for peeling+, but they do show in a graph that peeling+ gets a little over .8 precision while making guesses on almost all of the pairs, and we get a similar result. Note that with peeling+, recall and precision are the same except in the event that not all vertices are tie-broken. This

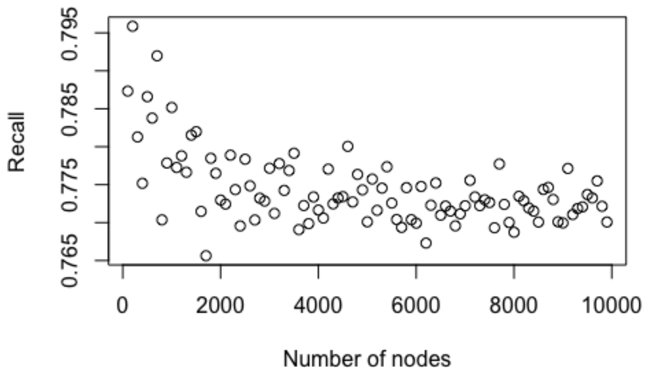

Figure 1: Recall for m = 5 on the peeling algorithm

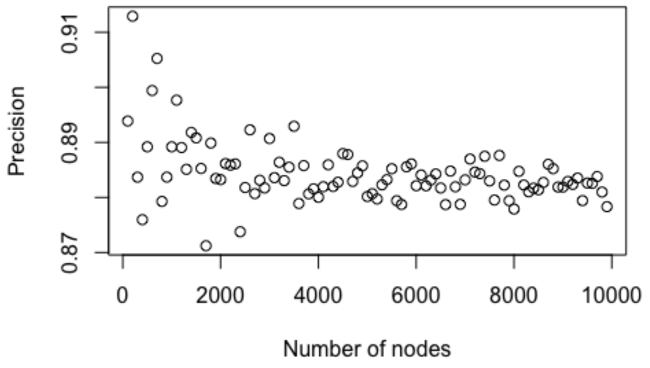

Figure 2: Precision for m = 5 on the peeling algorithm

happens, for example, in the case for $m = 1$ (the case of a tree), because the lack of edges makes it difficult to differentiate vertices. This is shown in figure 3. The measures prefixed with "aug" are the values for peeling+ and the measures prefixed with "orig" are the measures for peeling.
For other values of m, tiebreaking occurs well, as seen in the example for $m = 10$, which is in figure 4.

This finishes our attempts to reproduce the empirical results given in the Magner paper. Finally, we thought about trying to perhaps improve on the peeling algorithm by making a peeling algorithm that allows vertices to move between bins based on the age of their neighbors. If a vertex's neighbor bin numbers deviated enough (enough defined by a parameter) from its bin-mates, we would move that vertex up or down bins accordingly. In figure 5, we graph the results with variating parameters and find that the peeling algorithm is optimal over all parameters. Therefore, we find that trying to exchange vertices between bins in this manner is suboptimal empirically.

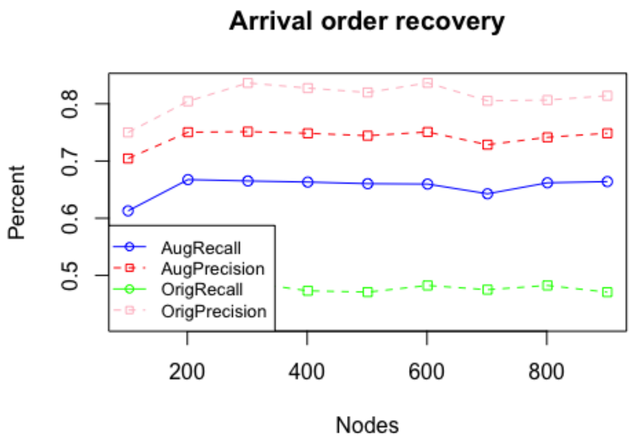

Figure 3: peeling+ vs peeling for m = 1

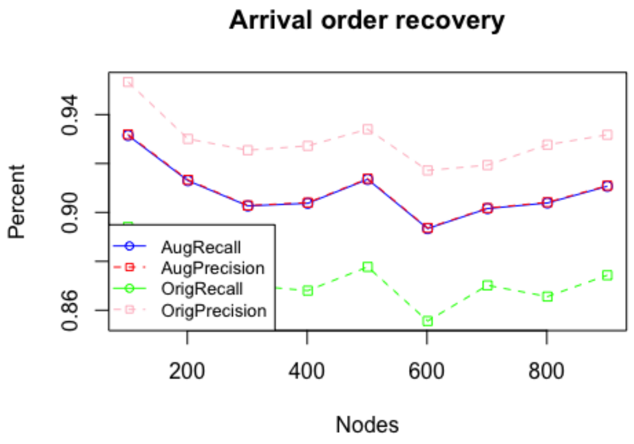

Figure 4: peeling+ vs peeling for m = 10

4.2 Interpolated Peeling

We want to try to find a lower bound for the precision curve. This means we want to find a relationship between recall and density. Magner et al. [1] have already found an upper bound which is reasonably tight. However, we want to try to find a lower bound for this curve. One way to do this is to simply analyze an algorithm and take the expectation of its measures of quality to get a lower bound on the precision curve (as then the optimal curve must perform better or equal to it). Precisely, we want to find f where

$$f(\theta(\sigma)) = f(E[\frac{\psi(\sigma)}{K(\delta)}]) = \delta(\sigma) \tag{5}$$

Where σ is generated according to whatever algorithm is used.

$$\sigma = \sigma(G, \mathscr{A}) \tag{6}$$

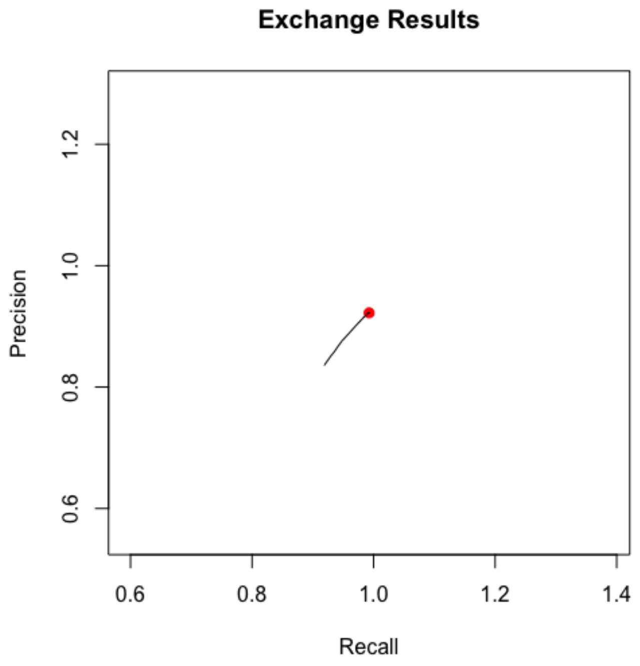

Figure 5: Graphing recall and precision at varying parameters for our algorithm that allows vertices freedom to move between bins. The red dot is the peeling algorithm. Clearly, the peeling algorithm is optimal. The parameters are m = 10 and n = 1000, but we also observe this for other settings for m and n.

Which expresses σ's dependence on both the graph G which is generated by the Barabasi-Albert process and the algorithm $\mathscr{A}$.

The algorithm that we want to analyze to try to characterize this lower bound is the interpolated peeling algorithm. For brevity, we will call it iPeeling. The algorithm runs as follows.

First, it runs the peeling algorithm. Then for each bin B, if the mean neighbor bin numbers of a particular vertex is an outlier (defined in the z-score sense) within a particular bin, it forms a new bin (with any ties that this outlier has) appropriately above or below the old bin. This process repeats for every bin until there are no outliers left in any bins. It is obvious that varying the z-score necessary to be considered an outlier can change how many bins are created and therefore how many guesses iPeeling makes. We (creatively) name this parameter z. We call the iPeeling algorithm with parameter z $iPeel(z)$.

Note that we automatically have

$$\psi(\sigma(G, iPeel(z)) - \psi(\sigma(G, peeling)) \geq 0 \ \forall z \tag{7}$$

because this difference is simply the extra number of guesses that $iPeel(z)$ gets correct, because $iPeel(z)$ is stable with respect to peeling, e.g. all guesses that peeling makes $iPeel(z)$ makes the same guesses (because iPeel(z) only tiebreaks within bins).

Naturally, we run some experiments on the iPeeling algorithm. First, as expected, its precision and recalls are sandwiched between the precision and recalls of peeling and peeling+. This is shown in figure

6. Next, we investigate the iPeeling algorithm over many different values of z. Note how linear the line

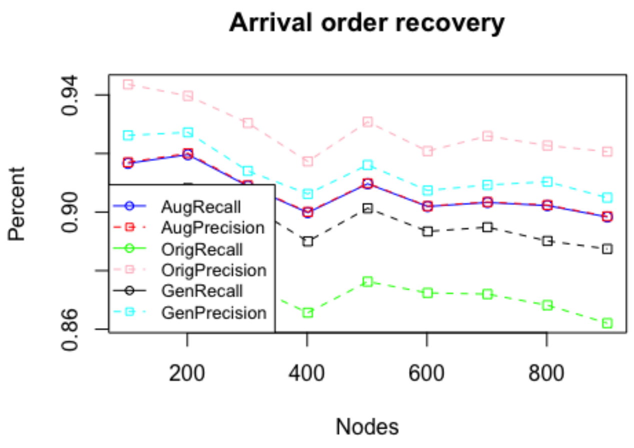

Figure 6: The measures prefixed with "Gen" are the ones made from iPeel. Note how its recalls and precisions are between the ones of peeling and peeling+.

going through the points is. Similar results were also found for other values of m. This empirically linear

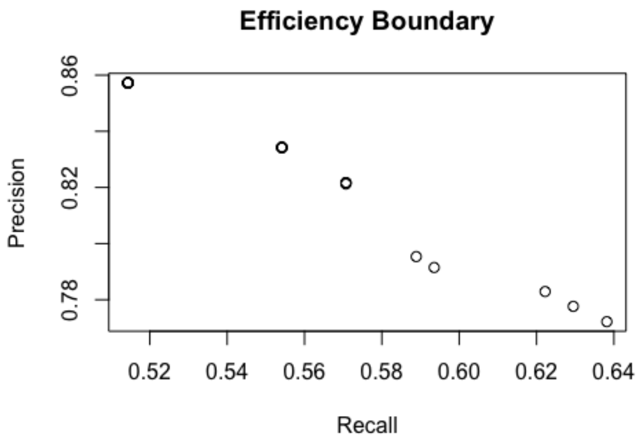

Figure 7: Each point represents a value of z given to the iPeel algorithm, for m = 1. Note that many points are stacked on top of one another (there are 100 points in this graph). This suggests it is difficult to access certain values for recall and precision using iPeel.

relationship gives some clues on future directions for theoretical research into this algorithm. However, before we can start to analyze the iPeeling algorithm, we have to understand theoretically how the peeling algorithm works. The following section addresses this.

4.3 Investigating the Peeling Algorithm

Let $G^{(t)}$ be a graph generated by the Barabasi-Albert process and fix it. Now bring in a new vertex, which forms edges with vertices in the graph according to the Barabasi-Albert process. Then we want to study the behavior of the vertices and their bins. Let $\{B_i^{(t)}\}_i$ be the ith bin, where the first bin is the bin with the vertices with the lowest degree. The first natural question to ask is, what is the behavior of $\{B_i^{(t+1)}\}$ as a result of introducing the new vertex? Understanding this is a basic prerequisite to understanding the measures of quality that the peeling algorithm outputs.

Definition 1 (Promotion): Let v be a vertex in $B_i^{(t)}$. v is said to be *promoted* at time $t+1$ if v is in B_{i+1}^{t+1}, e.g. it moves up a bin after a vertex is added to the graph. For brevity, whenever v is said to be promoted, it will be assumed that it means it is being promoted at time $t+1$.

Definition 2 (Bin path): A path from a vertex $v \in B_i^{(t)}$ to a vertex $v' \in B_1^{(t)}$ is a path between v and v' s.t. the next vertex along the path is a vertex in $B_{i-1}^{(t)}$, and the vertex after that is a vertex in $B_{i-2}^{(t)}$, and so on until the path reaches the vertex $v' \in B_1^{(t)}$. The "end" of the bin path is the vertex in the first bin.

Definition 3 ($deg_+^{(t)}(v)$) The number of edges going to vertices that are in older bins. Note that by Lemma 3.3 of [1] this is exactly m for vertices not in the oldest bin.

Definition 4 ($deg_i^{(t)}(v)$) The number of edges v has in $G^{(t)}$ after $G^{(t)}$ has gone through $i-1$ steps in the peeling process. Note that this is simply all the vertices in bins above or equal to the ith bin.

Lemma 1:
(1) if $v \in B_i^{(t)}$ is a part of a bin path, and if the vertex introduced in the next time step adds itself to the end of its bin path, then $v \in B_{i+1}^{t+1}$.
(2) if v is not part of a bin path, then it cannot increase in bin number.

Proof. We prove this by induction on i, the bin number.
<u>Base case:</u> i = 1
(1) If $v \in B_1^{(1)}$ is a part of a bin path, and if the vertex introduced in the next time step adds itself to the end of its bin path, then $v \in B_2^2$ because it now has $m+1$ edges.
(2) all the vertices are part of a bin path.

<u>Inductive case i + 1:</u>
Assume conclusion for i.
(1) If $v_{i+1} \in B_{i+1}^{(t)}$ is part of a bin path (and not a part of the oldest bin) then it must have an edge to $v_i \in B_i^{(t)}$ that is part of a bin path. If v_i were promoted, then because v_{i+1} has an edge with v_i,

$$deg_{i+1}^{(t+1)}(v_{i+1}) > deg_{i+1}^{(t)}(v_{i+1}) = m = deg_{i+1}^{(t)}(v_i) \tag{8}$$

Therefore, v_{i+1} must rise in bin number. Next, note that by inductive assumption (2) the only way to rise in bin number is through being part of a bin path, and by the inductive assumption (1) bin path promotion can only increase bin number by one, so no vertices below v_{i+1} can rise past v_{i+1}.

$$deg_{i+2}^{(t+1)}(v_{i+1}) \le deg_+^{(t)}(v_{i+1}) = m \le deg_{i+2}^{(t+1)}(v') \ \forall v' \in G_{i+2}^{(t)} \tag{9}$$

Where $G_{i+2}^{(t)}$ is $G^{(t)}$ after $i+1$ peeling steps. Therefore, v_{i+1} has the minimum degree in $G_{i+2}^{(t+1)}$, and thus would be placed in bin $i+2$.

(2) If $v_{i+1} \in B_{i+1}^{(t)}$ is not in a bin path, then no vertex from below with an edge to v_{i+1} can get to the same bin or above. So $deg_{i+1}^{(t+1)}(v_{i+1})$ does not increase and so v_{i+1} cannot rise. □

The next question to ask is what does the bin sizes actually look like at a given point in time? This is important, because knowing the bin sizes directly gives us the number of guesses the algorithm is making. Furthermore, it will give more clues on what the number of correct guesses is. We investigated this empirically and found that it is roughly exponential in the bin number, in figure 8. To try to show

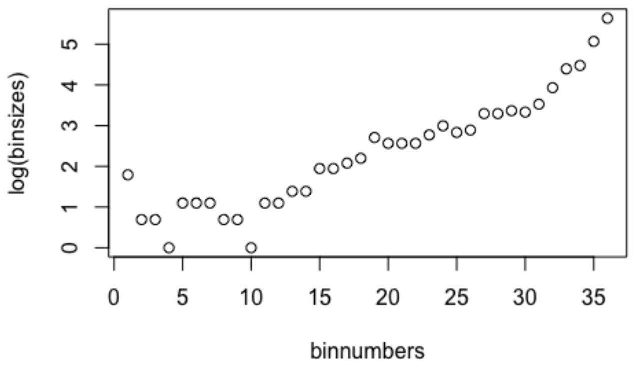

Figure 8: A log scaled graph of the bin sizes generated n = 1000 and m = 5. Note the mostly linear nature of the line. Ignore the beginning and ends of the graph, as they are usually very different from the general trend due to their corner cases.

that the distribution of the bin sizes is exponential in bin number (or some other distribution), we want to study $|B_i^{(t)}|$ as a process over t, where t is both the number of nodes as well as the notion of time. To do this, we need to understand probabilistically what happens to bin sizes when a vertex is added. And by Lemma 1, this is entirely determined by the lengths of the bin paths that the vertex introduced in the new time step forms an edge to. This, in turn, is determined by how many bins up vertices tend to have edges to (recall that a bin path requires that each vertex have an edge to a vertex above it).

We first investigate how many bins up vertices tend to have edges to empirically. Figure 9 shows one such distribution, and so does figure 10. The exponential trend suggests a geometric distribution,

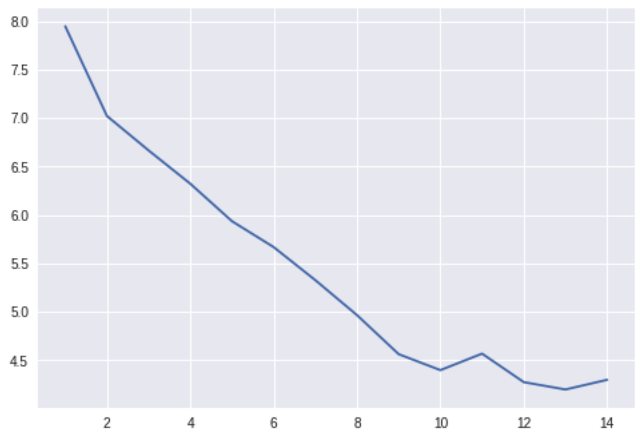

Figure 9: A log scaled graph of the edge distributions of vertices in the first bin generated n = 10000 and m = 1. This is averaged over 50 graphs. x-axis is how many bins up the edges are and y-axis is how many edges satisfy this. Note that this is linear away from the beginning and ends of the graph.

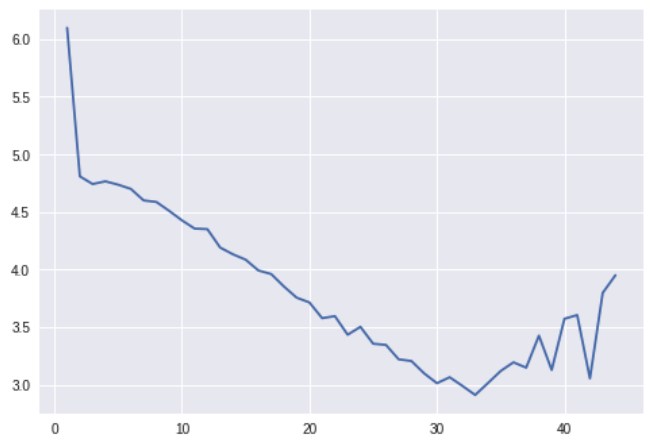

Figure 10: A log scaled graph of the edge distributions of vertices in bin 6 generated n = 10000 and m = 5. This is averaged over 50 graphs. x-axis is how many bins up the edges are and y-axis is how many edges satisfy this. Note that this is linear away from the beginning and ends of the graph.

where the parameters seem to depend on what m is. However, this is a little worrisome for our applications, because empirically the edge distribution to one bin above does not seem to follow a nice rule.

However, heuristically what we might be able to do is take the limit as n approaches infinity, and because infinite vertices should (loosely speaking) imply infinite bins, we won't have to worry about the right edge of the graph as much. Then one could take the complement of the cumulative distribution of the edge distribution past some number k, and this would give us the probability of a vertex having an edge to the k bins above it. If $k = 1$, then we can get exactly what we want: the proportion of edges going to the bin above. We sum up this intuition in the following conjecture:

Conjecture 1: $E[lim_{n\to\infty} \frac{|edges\ to\ bin\ i+j\ where\ bin\ j>t|}{|total\ number\ of\ edges\ from\ bin\ i|}] = ae^{-bj}$ where n is the number of nodes, a and b some constants.

Future directions for the research includes trying to show this conjecture rigorously, as well as use it to extend the theory so far.

5 Conclusion

This work makes some preliminary progress in theoretically understanding the peeling algorithm as well as the theoretical bounds on the feasibility of node age inference. We also introduce an algorithm that allows more control over the tradeoffs between recall and precision. As is evident from our analysis, there are a lot of moving parts to even when trying to analyze the peeling algorithm. Can iPeeling be rigorously studied enough to give some sort of lower bound? And can this lower bound prove some sort of optimality about the peeling, peeling+, and iPeeling algorithms?

6 References

[1] Magner, Abram N. and Sreedharan, Jithin K. and Grama, Ananth Y. and Szpankowski, Wojciech, TIMES: Temporal Information Maximally Extracted from Structures, In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, 2018

[2] Abram Magner, Ananth Grama, Jithin Sreedharan, and Wojciech Szpankowski. Recovery of vertex

orderings in dynamic graphs. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 2017.
[3] Christian Borgs and Michael Brautbar and Jennifer T. Chayes and Sanjeev Khanna and Brendan Lucier, The Power of Local Information in Social Networks, *CoRR*, abs/1202.6033, 2012
[4] Sbastien Bubeck and Luc Devroye and Gbor Lugosi, Finding Adam in random growing trees, *ArXiv e-prints*, arXiv 1411.3317, 2014
[5] Sbastien Bubeck and E. Mossel and Miklos Racz, On the influence of the seed graph in the preferential attachment model, *ArXiv e-prints*, arXiv 1401.4849, 2014