Recovering the first vertices in a Preferential Attachment model

Math 199H - Honors Thesis
Department of Mathematics,
University of California, San Diego

Spring 2020

Thu Nguyen
Advisor: Professor Ery Arias-Castro

June 5, 2020

Abstract

We study the problem of recovering the first vertices in a graph grown under the *preferential attachment* rules. We focus on tree graphs. We start with and build upon the algorithm introduced by Bubeck, Devroye, and Lugosi in [3] which looks at the largest connected component after removing any one vertex.

We first consider recovering only vertex 1. We provide a more detailed proof of the accuracy of the algorithm, which is taken from [3]. We test the algorithm with simulations over a wide range of graph sizes. Given the computation cost, we propose a two-step procedure which combine that with using the vertex degree to cut down the complexity without giving up much accuracy. We then consider recovering the first L vertices. We propose a sequential algorithm where at each stage we do a local search conditioned on what already know. We conjecture on the expectation of that algorithm and then test the algorithm on simulations.

We find that we can get significantly accurate when recovering vertex 1. However, the problem starts becoming much harder once we want to recover more vertices. In particular, it only takes until the fifth vertex to observe considerable drops in the accuracy. Nonetheless, it appears that it is possible to increase the accuracy greatly if we are willing to return a larger set of suspected vertices.

Acknowledgement

I would like to express the most sincere thanks to Professor Arias-Castro, who offered me the opportunity to work on this project, provided resources, and guided me through how to tackle such open problems. It was an eye-opening experience for me.

Contents

1 Introduction

Preferential attachment model was first introduced in the 1970s by Price in [12] and has picked up in popularity since the 1990s after the paper by Barabási and Albert in [1]. It is one of the simplest models that exhibits some important features of real networks, ranging from biology and transportation networks, and the classic example of citation networks.

One reason for why the *preferential attachment* model is commonly observed relies on the way *preferential attachment* model grows, which mimics the phenomena of *the rich get richer*: when graphs evolve over time, newly added vertices will have tendency to connect to already well-connected vertices.

We introduce a (somewhat) standard version of a *preferential attachment* mode. This takes three parameters, namely:

1. G_0 : the seed graph at time $t = 0$
2. m : the number of new edges added at each iteration;
3. α : the parameter which controls the growth of the graph.

We can now define the graph iteratively as follows. Let G_0 be given. Let $d_t(u)$ be the degree of vertex u in $G(t)$. We construct $G(t+1)$ as:

1: Add a new vertex $t+1$
2: Choose m existing vertices $u \in G(t)$ such that

$$\mathbb{P}(u \text{ is chosen}) \ \propto \ \big(d_t(u)\big)^{\alpha} \tag{1}$$

3: Add new edges between $t+1$ and the m chosen vertices

The specification above leaves us a lot of freedom to tweak the model. Some choices we can make include:

1. m : the larger the m, the denser the graph, but also more variation between different realizations of such a model;
2. α : the larger the α, the faster the already well-connected vertices will be even more connected, the behavior of which closely mimics the observed real-life phenomena of *the rich get richer*; common choices of α are 0 and 1;
3. if $m > 1$, we can either choose m vertices sequentially or simultaneously, the former of which further enhances the *the rich get richer* phenomena;

4. we can introduce another parameter, say $p \in [0, 1]$: a *Bernoulli* random variable which decides if at the current iteration we want to add a new vertex and then new edges or we can just add new edges between the vertices.

A defining characteristic of the *preferential attachment* model is its degree profile, which closely follows a *power-law*. This characteristic has been studied rigorously and in great detail, such as by Dorogovtsev et al in [5]. Many of those results have been summarized in the recent book on random graphs by Hofstad in [8]. In particular, given a strict *preferential attachment* model (with $m = 1$ and $\alpha = 1$), it has been shown, for example by Chung and Lu in [4] and Bollobas et al in [2], that as t goes to infinity, as k gets large, the proportion of vertices with degree equal to k is proportional to k^{-3}. We observe that the older a vertex, the larger the expected degree of that vertex. Figure 1 shows an instance of such model. We observe a few vertices with high degrees while the majority are of degrees 1 and 2.

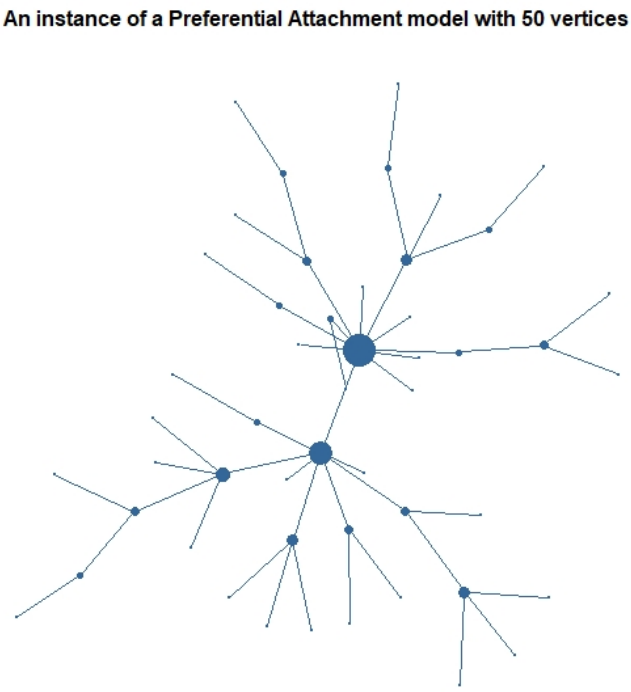

Figure 1: An example of a *preferential attachment* model starting from a single vertex with α equal to 1, and m equal to 1 (one new edge at each iteration).

2 Overview

The motivation for our study is to recover the first vertices of a graph assumed to have grown under a *preferential attachment* mechanism.

Suppose we have a graph G, which we believe grew according to our *preferential attachment* model specified in 3.1.1. G gives us a snapshot of our graph at some point in time. It is natural to ask ourselves if we could trace back in time and recover graphs at previous past. Of particular interest is the seed graph which gave rise to G. In other words, we believe that G started with a single vertex, say 1, we now want to identify that vertex.

We first consider the special case of recovering vertex 1, and then generalize that to recover first L vertices. To introduce a bit of structure to the graphs, we work with *preferential attachment* model on tree, which will be defined later on. Section 3 addresses the problem of recovering vertex 1, and section 4 that of L vertices.

Inspired by the algorithm and their results on the accuracy of such algorithm introduced in the paper by Bubeck, Devroye, and Lugosi [BDL [3]], within each problem, we first define the set-up and the objectives, together with other considerations, such as the computation cost or the need to be accurate versus precise. We next introduce or propose algorithms to tackle the problems. For each algorithm, we introduce the motivation and the theoretical results or supporting arguments. We then test the algorithms over a wide range of graph sizes and note the results.

Furthermore, to keep track of the simulation performance on vertex recovery, we introduce two performance metrics:

$$\begin{aligned} \textit{Accuracy} &= \text{the probability of fully recovering the first } L \text{ vertices} \\ \textit{Precision} &= \frac{L}{K} \end{aligned}$$

Ideally, we look for an algorithm that could be both highly accurate and highly precise. It should be noted that as introduced by Lugosi and Pereira in [10], in the problem of recovering the first L veritces, there are two notions of *accuracy* of an algorithm: suppose the algorithm returns a set of vertices, say H, then:

- *acuracy of first kind:* $H \subseteq L$
- *acuracy of second kind:* $L \subseteq H$

In this study, we will focus the accuracy of *second kind*, ie. we want to fully recover the first L vertices.

As a note, we assume that the reader is familiar with basic concepts in *graph theory*, such as

- simple graph, connected graph, tree graph
- vertex degree, vertex neighbor

For additional references, the reader can consult *Networks* by Newman [11].

Notation 2.1. *Let G be a fully labeled graph, then G^o denote the isomorphism class of G, ie. G^o is an unlabeled copy of G. An example is in figure 2.*

Notation 2.2. *Let $L \in \mathbb{N}$, we denote $[L] = \{1, 2, \ldots, L\}$.*

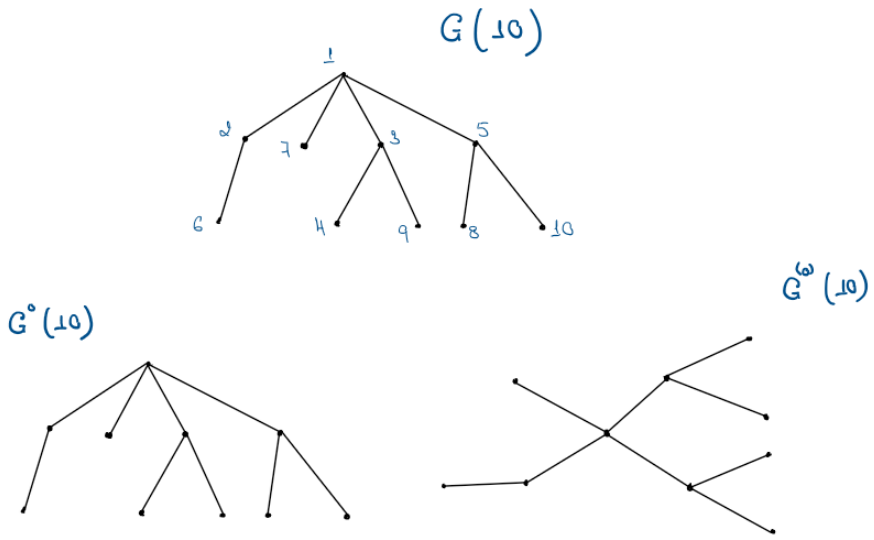

Figure 2: Example of a fully and chronologically labeled graph versus its unlabeled copies in different orientations.

3 Recovering Vertex 1

3.1 Set-up and Objectives

3.1.1 Set-up

We will start by specifying the exact settings and the objective. We consider a special case of a *preferential attachment* model introduced in section 2, one in which $m = 1$ and $\alpha \in \{0, 1\}$. These choices of parameters give a *uniform attachment* and *preferential attachment* on trees respectively.

Additionally, we assume that the seed graph G_0 is a singleton (a single vertex with no edges).

The growth rule in 1 implies that

(a) when $\alpha = 0$, we have the *uniform attachment* model:

$$\mathbb{P}(u \text{ is chosen}) \;=\; \frac{1}{|V\big(G(t)\big)|} \;=\; \frac{1}{t}$$

(b) when $\alpha = 1$, we have the *preferential attachment* model:

$$\mathbb{P}(u \text{ is chosen}) \;=\; \frac{d_t(u)}{\sum\limits_{v \in V(G(t))} d_t(v)} \;=\; \frac{d_t(u)}{2(t-1)} \qquad (\text{if } t > 1)$$

Figures 3 and 4 illustrate two instances of the evolution of the *uniform attachment* and *preferential attachment* models respectively at sizes $20, 60$, and 100, starting from a singleton and growing iteratively.

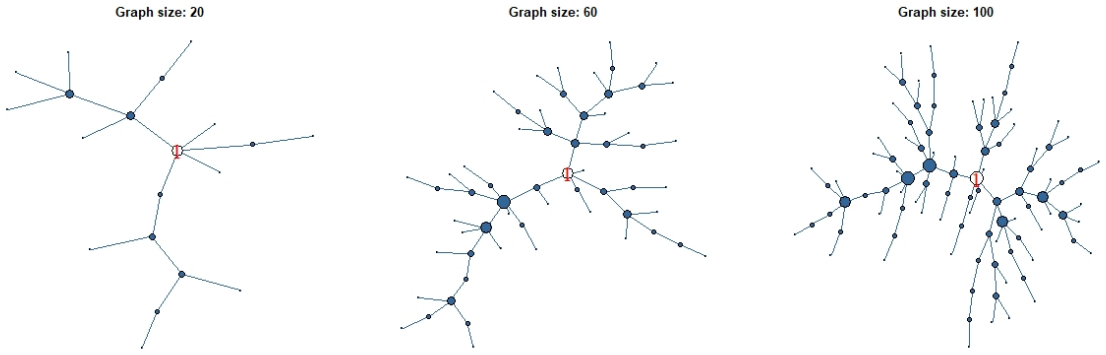

Figure 3: Evolution of an instance of the *uniform attachment* model, starting with a single vertex 1, which is labeled and colored to give us a sense of where the vertex is relative to the whole graph.

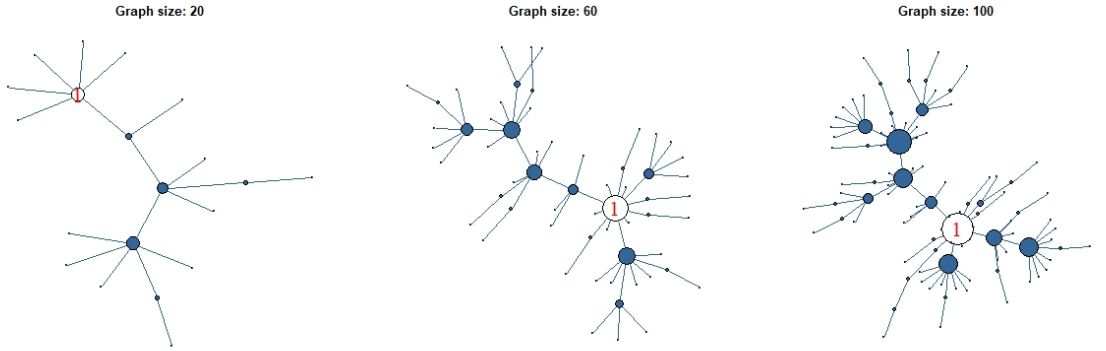

Figure 4: Evolution of an instance of the *preferential attachment* model, starting with a single vertex 1, which is labeled and colored to give us a sense of where the vertex is relative to the whole graph.

3.1.2 Objectives

Suppose we have a graph G, which we believe grew according to one of our *preferential attachment* models specified in 3.1.1. G gives us a snapshot of our graph at a point in time. It is natural to ask ourselves if we could trace back and recover graphs at previous past. Of particular interest is the very first evolutions of the graph which gave rise to G.

In this section, we believe that G started with a singleton, ie. G_0 consists of only vertex 1. We now want to identify that vertex.

Not only do we want to identify vertex 1, we want to be as *accurate* and *precise* as possible, which were defined in section 2. We now establish certain criteria that we would like the algorithms to achieve. The first is on the accuracy:

> Fix some error rate $\varepsilon > 0$. We want an algorithm that takes in an unlabeled graph G and returns K vertices such that:
>
> $$\mathbb{P}(1 \in K) \;\geq\; 1 - \varepsilon$$

The second is on the precision:

> In returning K vertices, for a graph G of n vertices, given fixed error rate of ε, we want K as small as possible.

It turns out that in this context it is indeed possible to return only finitely many vertices even when the graph size goes to infinity, as proved in [BDL [3]]. The details of the algorithm will be discussed in section 3.2.

We can now formalize the problem. To differentiate a fully chronologically labeled graph of n vertices, $G(n)$, from its unlabeled copy, let $G^o(n)$ be its unlabeled tree graph. Assume $G^o(n)$ grew under the model specified in 1. Let $\varepsilon > 0$ be given. We look for a map H which takes in $G^o(n)$ and returns a subset of vertices in $G^o(n)$ such that

$$\lim_{n\to\infty} \mathbb{P}\Big(1 \in H\big(G^o(n)\big)\Big) \;\geq\; 1 - \varepsilon$$

3.2 Algorithm

As introduced in section 1, there is a strong positive relationship between the vertex age and its degree. In particular, as t goes to infinity, as k gets large, the proportion of vertices with degree equal to k is proportional to k^{-3}.

Given the observations, it is natural to think of using the vertex degree as the criteria in returning those K vertices. However, it is claimed in [BDL [3]] that to guarantee the accuracy, we would need to return $\mathcal{O}(\ln(n))$ vertices, where n is the number of vertices in the graph G. In particular, when n goes to infinity, we would need to return infinitely many vertices to be guaranteed of its accuracy.

In the same paper, Bubeck, Devroye, and Lugosi introduce a different algorithm. Instead of looking at the degree of a vertex u, they look at the size of the maximum connected component. We will sometimes refer to this as the φ algorithm or the φ method interchangably. We first introduce a notation.

Notation 3.1. *$(T, u)_{v\downarrow}$ is the subtree rooted at v, a child of u, and not containing u. Figure 5 illustrates an example of this.*

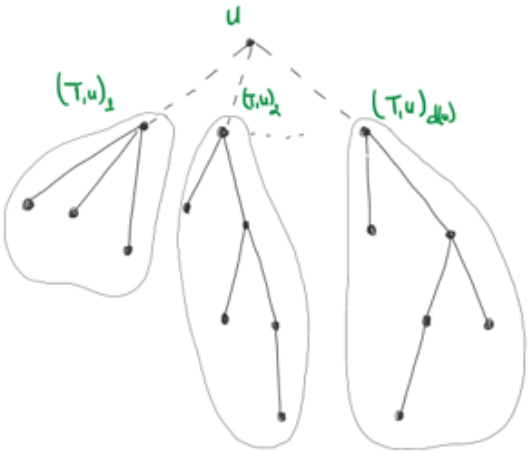

Figure 5: An example of $(T, u)_{v\downarrow}$, where v is numbered $1, 2, \ldots, d(u)$.

Since we only consider tree graphs, we will use G and T interchangably. We are now ready to describe the algorithm. For simplicity of notation, for the purpose of describing algorithms, let $G(n)$ denote an unlabeled graph of size n, and $N(u)$ be the set of vertices neighboring vertex u.

Algorithm 1 Recovering vertex 1, [BDL [3]]

Input: an unlabeled tree graph $G(n)$, K
for each $u \in V(G(n))$ **do**
    for each $v \in N(u)$, a neighbor of u **do**
        Compute $|(G,u)_{v\downarrow}|$
    end for
    Return $\varphi(u) = \max_{v \in N(u)} \left|(G,u)_{v\downarrow}\right|$
end for
Output: K vertices with the smallest φ values

Figure 6 gives a simple illustration of the algorithm.

Figure 6: The first graph is complete with 50 vertices. In the top right graph when removing a vertex u, the result implies that $\varphi(u) = 26$. Similarly, the bottom left implies that $\varphi(v) = 42$. The last is when we remove a leaf vertex w, (since the result is one connected component), hence $\varphi(w) = 49$. We now have an increasing order of $\{\varphi(u), \varphi(v), \varphi(w)$. If we choose to return $K = 1$, the algorithm will return vertex u, among the three of them.

We present here the results on the asymptotic accuracy of the algorithm on the *preferential attachment* and *uniform attachment* models which start from a singleton and grow according to growth rule 1.

Theorem 3.1: Uniform attachment model on tree, [3]

Consider the *uniform attachment* model starting with a singleton and has n vertices, denoted by $T(n)$. Let $T^o(n)$ be its unlabeled copy. Let $K \geq 2.5\frac{\log(1/\varepsilon)}{\varepsilon}$, then

$$\liminf_{n\to+\infty} \Big(1 \in H_\varphi\big(T^o(n)\big)\Big) \;\geq\; 1 - \frac{4\varepsilon}{1-\varepsilon}$$

We first give a sketch of the proof here.

1. Show that: $\mathbb{P}(1 \notin H_\varphi) \;\leq\; \mathbb{P}\Big(\varphi(1) \geq (1-\epsilon)n\Big) \;+\; \mathbb{P}\Big(\exists i > K : \varphi(i) \leq (1-\epsilon)n\Big)$
2. Show that: $\limsup_{n\to+\infty} \mathbb{P}\Big(\varphi(1) \geq (1-\epsilon)n\Big) \leq 2\epsilon$
3. Show that: $\limsup_{n\to+\infty} \mathbb{P}\Big(\exists i > K : \varphi(i) \leq (1-\epsilon)n\Big) \;\leq\; K(1-\epsilon)^{K-1}$

Remark: contrary to the procedure described in the algorithm above, where we were working with an unlabelled graph, here, to analyze the accuracy, we work with a chronically labeled graph, i.e. we know the ground truth: the exact order of which vertices appeared.

Proof: We now give a complete proof.

Let $T_{i,k}$ $(i \leq k)$ be the tree containing vertex i by removing from $T(n)$ all the edges between vertices $\{1, 2, \ldots, k\}$. At time k, this gives us a collection of k singletons.

Given a new vertex t, since t is equally likely to be connected to any existing vertex, from the perspective of growing subtrees $T_{i,k}$, the probability of t getting added to a particular subtree $T_{i,k}$ is proportional to the current size of $T_{i,k}$. Hence, growing $T_{i,k}$ is equivalent to the classic Polya Urn Model with k bins starting with 1 ball in each bin.

Let $\frac{|T_{i,k}|}{n}$ be the proportion of the size of bin i relative to all k bins. As n goes to infinity, we have:

$$\left(\frac{|T_{1,k}|}{n}, \frac{|T_{2,k}|}{n}, \ldots, \frac{|T_{k,k}|}{n}\right) \xrightarrow[n\to\infty]{\mathbb{P}} Dir(1, 1, \ldots, 1) \tag{2}$$

Further information on the derivation and details on Dirichlet distribution can be found in the paper by Frigyik et al at [7].

1. We first note that the event of vertex 1 not recovered implies that there must be at least one vertex $i > K$ such that $\varphi(i) \leq \varphi(1)$, which gives:

$$\begin{aligned} 1 \notin H_\varphi \;&\subseteq\; \exists i > K : \varphi(i) \leq \varphi(1) \\ \implies \mathbb{P}\left(1 \notin H_\varphi\right) \;&\leq\; \mathbb{P}\left(\exists i > K : \varphi(i) \leq \varphi(1)\right) \\ \implies \mathbb{P}\left(1 \notin H_\varphi\right) \;&\leq\; \mathbb{P}\left(\varphi(1) \geq (1-\varepsilon)n\right) + \mathbb{P}\left(\exists i > K : \varphi(i) \leq (1-\varepsilon)n\right) \\ &\qquad\qquad \text{(by the union bound)} \end{aligned}$$

2. By construction, $\varphi(1)$ is the size of the largest subtree starting at a child of 1.

   Now, recall the definition from step 1, $T_{1,2}$ and $T_{2,2}$ are the two subtrees each starting at 1 and 2 after removing the edge $\{1, 2\}$. Since $T_{1,2}$ always includes vertex 1, when computing any subtrees starting from children of 1 given $T_{1,2}$, we always have $\varphi(1) < T_{1,2}$. Hence:

$$\varphi(1) \;\leq\; \max\left\{|T_{1,2}|,\; |T_{2,2}|\right\}$$

   Note that the equality sign happens when 1 is a leaf. Furthermore, note that after time $t = 2$, in which case we have 2 vertices, it follows that vertex 3 has equal and identical chances of connecting to either vertex. By symmetry:

$$i \in \{1, 2\} : \quad \frac{|T_{i,2}|}{n} \xrightarrow[n\to\infty]{\mathbb{P}} Unif[0, 1]$$

   We note that this derivation is a special case of 2. In particular, they are identically distributed and hence

$$\limsup_{n\to+\infty} \mathbb{P}\left(\varphi(1) \geq (1-\epsilon)n\right) \;\leq\; \limsup_{n\to+\infty} \mathbb{P}\left(\frac{|T_{i,2}|}{n} \geq (1-\epsilon) \;\middle|\; i \in \{1, 2\}\right) \;=\; 2\epsilon$$

3. Let i be a vertex which was added after the first K vertices were added. We observe that from removing the path between i and those K vertices the inequality

$$\varphi(i) \;\geq\; \min_{1\leq k\leq K} \sum_{j=1, j\neq k}^{K} |T_{j,K}|$$

   Recall that the vector $\frac{1}{n}(|T_{1,K}|, \ldots, |T_{K,K}|)$ converges in distribution to a Dirichlet distribution with parameters $(1, \ldots, 1)$, it follows that:

$$\frac{1}{n} \sum_{j=1, j\neq k}^{K} |T_{j,K}| \xrightarrow[n\to\infty]{\mathbb{P}} Beta(K-1, 1) \tag{3}$$

   More information on the Beta distribution can be found in any standard textbook on probability, such as [13].

As a result,

$$\begin{aligned}
&\limsup_{n\to+\infty} \mathbb{P}\left(\exists i > K : \varphi(i) \leq (1-\epsilon)n\right) \\
&\leq \limsup_{n\to+\infty} \mathbb{P}\left(\exists k \in [K] : \min_{1\leq k\leq K} \sum_{j=1, j\neq k}^{K} |T_{j,K}| \leq (1-\epsilon)n\right) \\
&\leq K(1-\epsilon)^{K-1}
\end{aligned}$$

Let $K \geq 2.5\frac{\log(1/\varepsilon)}{\varepsilon}$. From steps $2, 3, 4$ above, we have:

$$\begin{aligned}
\mathbb{P}(1 \notin H_\varphi) &\leq \mathbb{P}\Big(\varphi(1) \geq (1-\epsilon)n\Big) + \mathbb{P}\Big(\exists i > K : \varphi(i) \leq (1-\epsilon)n\Big) \\
&\leq 2\epsilon + K(1-\epsilon)^{K-1}
\end{aligned}$$

and the result follows. ■

Theorem 3.2: Preferential attachment model on tree, [3]

Consider the *preferential attachment* model starting with a singleton and has n vertices, denoted by $T(n)$. Let $T^o(n)$ be its unlabeled copy. Let $K \geq C\frac{\log^2(1/\varepsilon)}{\varepsilon^4}$ for some constant $C > 0$, then

$$\liminf_{n\to+\infty} \Big(1 \in H_\varphi\big(T^o(n)\big)\Big) \geq 1 - \varepsilon$$

The proof of this result follows the same outline as given at the beginning of the proof of theorem 3.1. We note here the key difference.

Assume the same definition of $T_{i,k}$. Whereas in Theorem 3.1, $\left(|T_{1,k}|, \ldots, |T_{k,k}|\right)$ follows a standard Polya urn model with k bins, here, $2\left(|T_{1,k}|, \ldots, |T_{k,k}|\right)$ follows a Polya urn model with k bins and replacement matrix $2I_k$ (a replacement matrix simply tracks the number of new balls to be added to each bin at each iteration). Further details on this variant can be found in [9] by Janson.

As a result, conditioned on the graph $T(k)$, we have the convergence

$$\left(\frac{|T_{1,k}|}{n}, \frac{|T_{2,k}|}{n}, \ldots, \frac{|T_{k,k}|}{n}\right) \xrightarrow[n\to\infty]{\mathbb{P}} Dir\left(\frac{d_{T(k)}(1)}{2}, \ldots, \frac{d_{T(k)}(k)}{2}\right) \tag{4}$$

where $d_{T(k)}(u)$ is the degree of vertex u in the graph $T(k)$.

3.3 Simulation

We would like to study the algorithm via simulations. In particular, the focus is on

1. how fast the accuracy converges, if any, and
2. how many vertices need recovered to achieve significant accuracy

3.3.1 Simulation Data

We will build graphs of sizes over the range of:

$$n \in \{5000k : \ k \in [16]\}$$

(from 5000 to 80,000, in increments of 5000). In each graph size, we will build 1000 simulations. We will then run the algorithms over those 1000 simulations, and calculate the number of times 1 is recovered. We will do that over different choices of K:

$$K \in [10] \cup \{5k : \ k \in [20]\}$$

We will repeat the same procedure for two sets of graphs: one generated by a *uniform attachment* model on tree, and one by a *preferential attachment* model on tree.

Disclaimer: Due to the significant cost of implementing algorithm 1, we will only compute the φ value of the first 5000 vertices in all the simulations. However, based on subsequent tests, we do not expect the results would be significantly different if we computed the φ value on all the vertices.

3.3.2 Simulation Results

We first note that in the analysis, we use the terms *accuracy* and *recovery rate* (which is the ratio of times vertex 1 is recovered over the number of simulations) interchangably. We will focus on the results on the *preferential attachment* model.

Figure 7 shows the recovery rate across different value for K on *preferential attachment* model on trees of size $50,000$, via the method of minimum maximum subgraph. We have some observations:

1. The accuracy converges very quickly.
2. We only need to return 20 vertices to achieve an accuracy of at least 90%.
3. If we only take the vertex with the minimum φ value, we would be correct more than one third of the time, which increases to over two thirds with only three such vertices.

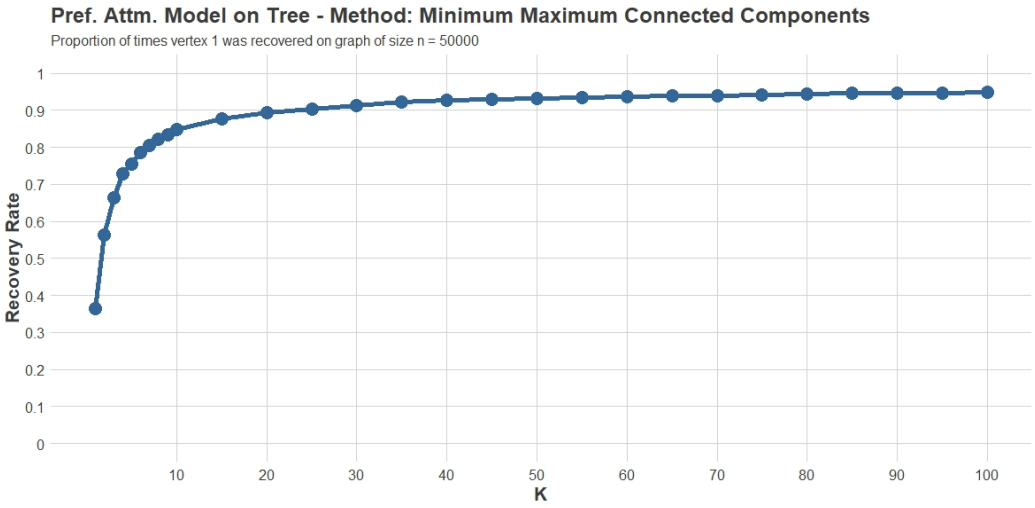

Figure 7: Recovery rates of vertex 1 on *preferential attachment* model on tree across different K on graphs of size $50,000$.

Figure 8 shows the long-run behavior of the recovery rate when the graph size increases while returning only 20 vertices. We have some observations:

1. Except for the rugged segment between sizes of $5,000$ to $15,000$, the recovery rate changed minimally even after the size increased by over 10 times.

2. The accuracy consistently stayed above 90% regardless of the graph sizes.

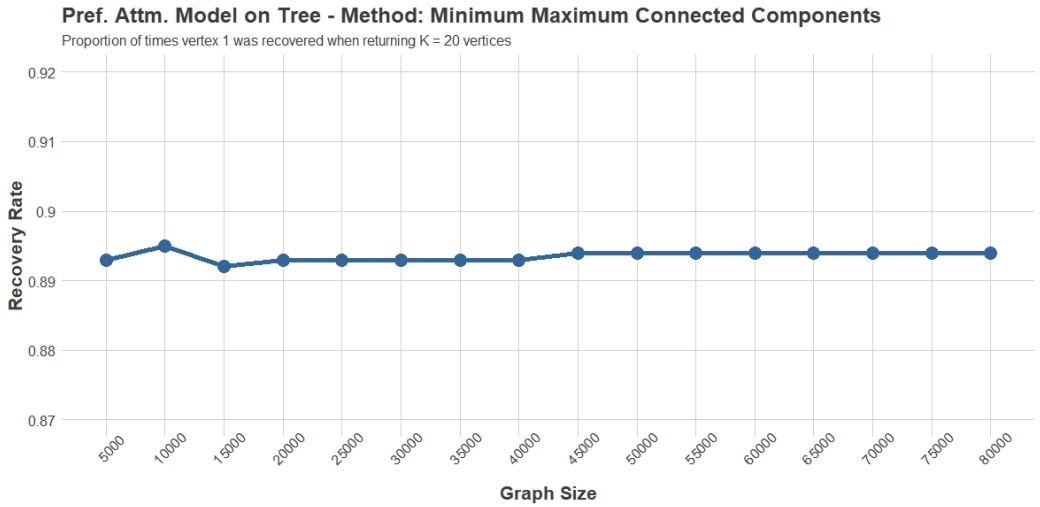

Figure 8: Recovery rates of vertex 1 on *preferential attachment* model on tree across different graph sizes when returning $K = 20$ vertices.

Figures 7 and 8 give us an impression that the algorithm is very robust: it can be precise and concise very quickly. Figures 9 and 10 show the recovery rates when we return only vertices with the smallest φ values. One observation significantly stands out:

1. The recovery rates remained unchanged across a wide range of graph sizes.

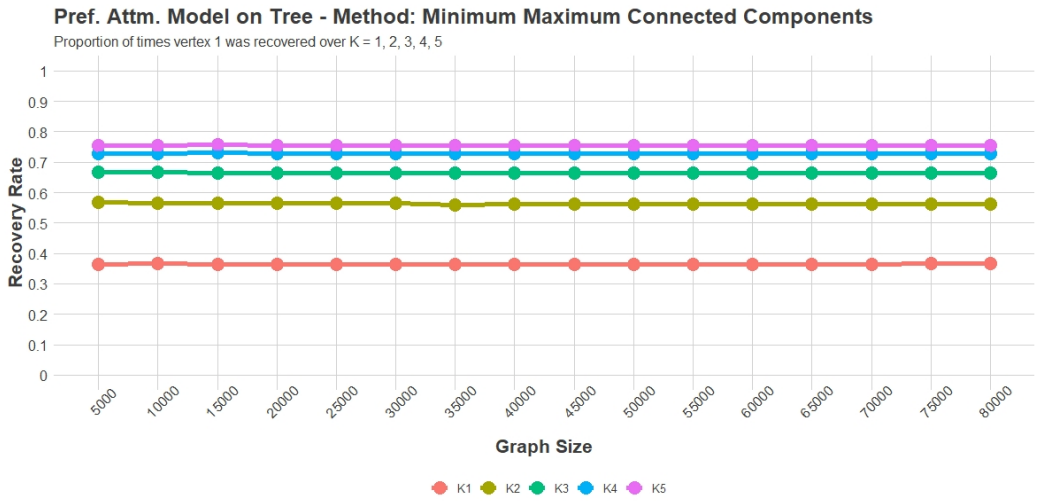

Figure 9: Recovery rates of vertex 1 on *preferential attachment* model on tree across different graph sizes when returning only $1, 2, 3, 4$ or 5 vertices.

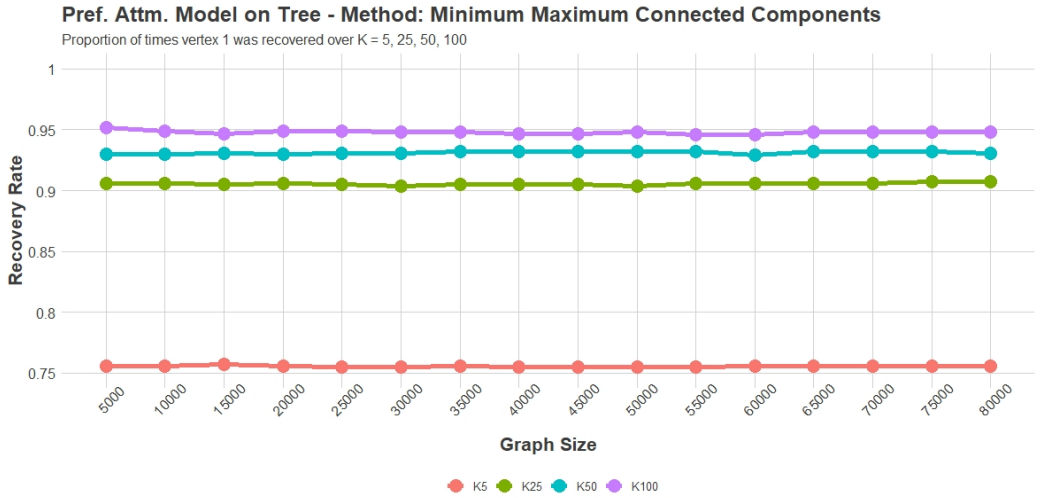

Figure 10: Recovery rates of vertex 1 on *preferential attachment* model on tree across different graph sizes when returning $5, 25, 50$ or 100 vertices.

We would like to now compare the two methods:

$$\underbrace{\text{minimum maximum connected component}}_{\text{method } (A)} \quad vs. \quad \underbrace{\text{maximum degree}}_{\text{method } (B)}$$

Figure 11 gives a look at how the two methods fare. We have some observations:

1. The naive approach of returning vertices of maximum degree yielded recovery rates that closely resembled that of the φ method.

2. Across all sizes, the accuracy from method (A) was strictly better than that of method (B).

3. Although less accurate, the accuracy gaps between the two methods were relatively small, at most 15% when we wanted to return less than 5 vertices, but reduced to less than 5% as soon as we were willing to return 50 vertices or more, as seen in figures 12 and 13.

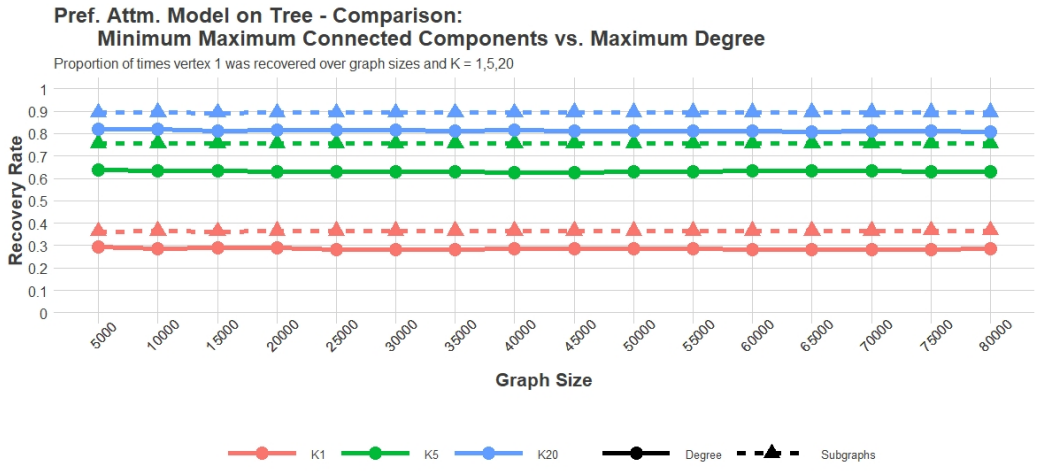

Figure 11: Recovery rates of vertex 1 on *preferential attachment* model on tree across different graph sizes when returning $1, 5$, or 20 vertices over the two methods.

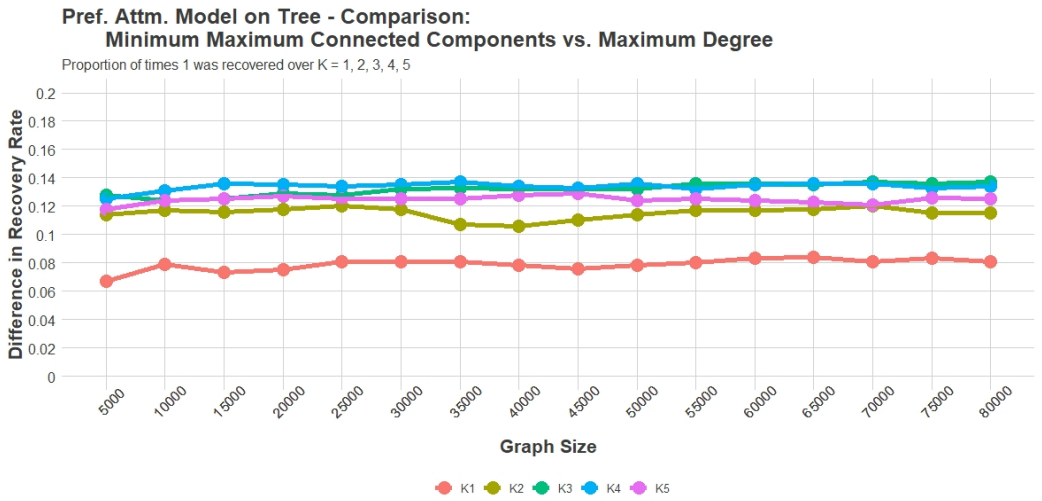

Figure 12: Differences in the recovery rates of vertex 1 on *preferential attachment* model on tree between the two methods when returning only $1, 2, 3, 4$ or 5 vertices.

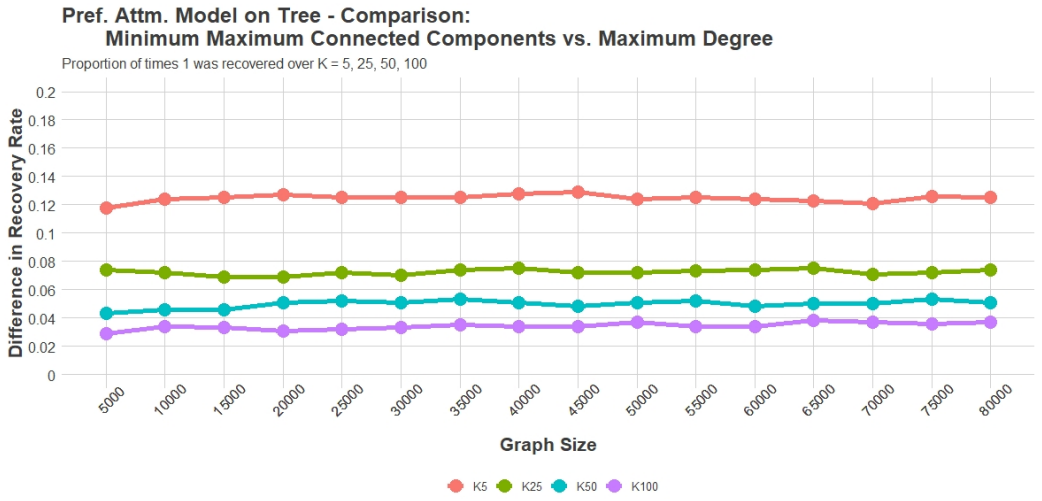

Figure 13: Differences in the recovery rates of vertex 1 on *preferential attachment* model on tree between the two methods when returning only $5, 25, 50$ or 100 vertices.

3.4 Proposed Variant

Inspired by the analysis, we have two observations regarding the two methods of φ versus degree.

1. Their accuracy rates closely resemble each other.
2. Method (A) can get very expensive compared to method (B), indeed while computing vertex degree can be done in real time, computing φ value even for only $5,000$ vertices per simulation can take more than 24 hours when the sizes get above $70,000$ for each batch of $1,000$ simulations for instance.

As such, we desire to keep the accuracy while bringing down the computation cost. We propose a combination of both methods.

We consider a 2-step procedure, in which we first look at M vertices with the maximum degree, to narrow down the set of vertices on which we now compute the φ value. We can describe the algorithm as:

Algorithm 2 Recovering vertex 1: Variant 1

1: Input: an unlabeled tree graph $G(n)$, K, M
2: Step 1: Return M vertices with the maximum degree
3: Step 2: Compute φ values among those M vertices
4: Output: K vertices with the smallest φ values

We now run the proposed algorithm on the same set of data. Figure 14 shows the result. We have some observations:

1. There appeared no significant (or even observable) changes in the accuracy when we computed the φ values on only a subset of vertices compared to that on all the vertices.

The simulation gives us support for our proposed variant, that to save computation time, we can incorporate the naive method of looking at the vertex degree to narrow down the set of suspected vertices.

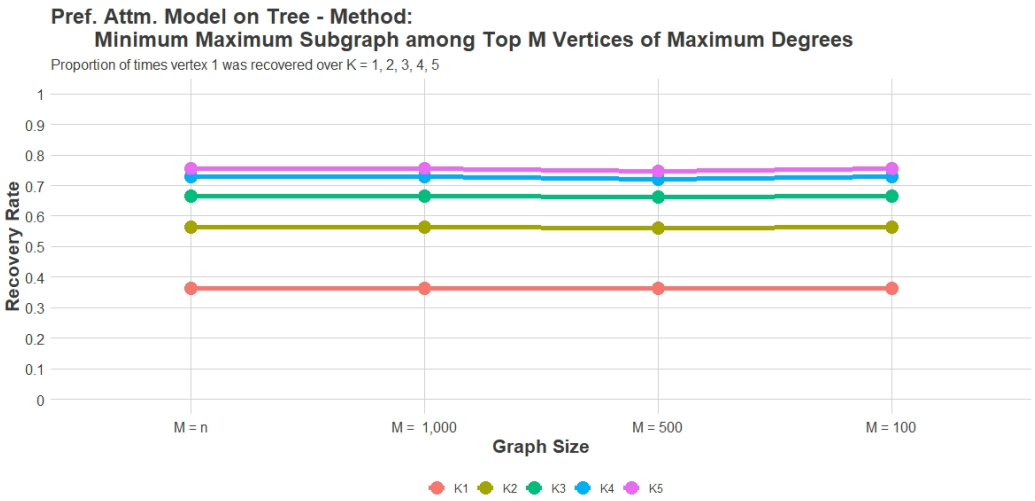

Figure 14: Recovery rates of vertex 1 on *preferential attachment* model on tree under the proposed 2-step algorithm to narrow down the set of vertices we want to compute φ, and returning only $1, 2, 3, 4$, or 5 vertices.

4 Recovering First L Vertices

4.1 Set-up and Objectives

4.1.1 Set-up

We follow the same set-up introduced in section 3.1.1. In particular, we focus on the *preferential attachment* model on tree.

4.1.2 Objectives

Generalizing the idea of recovering vertex 1, we would like to now recover the first L vertices. Ideally, we look for an algorithm that could be as accurate and precise as possible, in the hope of achieving performances similar to what we see in section 3.3.2.

4.2 Algorithm

A natural first guess is to simply apply algorithm 3.2 and possibly return more vertices. At the same time, a different and common approach is to do a local search conditioned on what we know at the current stage. The idea is popular, such as by Frieze and Pegden in [6], where the authors propose using local search starting at an arbitrary vertex u to recover the vertex 1. We will apply the idea but to recover L vertices.

We propose a *sequential local search* algorithm in stages, where at each stage we consider only the set of neighboring vertices of the current known vertices and then update those known vertices.

The algorithm can be described iteratively as:

Algorithm 3 Recovering first L vertices via local search

```
Input: an unlabeled tree graph G(n), K, L
Initialize a list M = ∅ by default
for each l ∈ 1 : L do
    if M = ∅ then
        Return K vertices with the smallest φ values
    else
        Create X containing neighbors of all vertices in M
        Choose from X K vertices with the smallest φ values
    end if
    Update M with those K vertices returned
end for
Output: M vertices
```

In other words, we start by recovering vertex 1. Having returned a set of vertices which we believe contain 1, denoted by K_1, we look at the vertices' neighbors, and choose from them another set of vertices with the smallest φ values, denoted by K_2. We note that $K_1 \cap K_2 = \emptyset$, and we believe in the likelihood that $\{1, 2\} \subset K_1 \cup K_2$. To recover subsequent vertices, we repeat the same procedure.

Let $G(n)$ be a tree graph with n vertices. We introduce some notations:

Notation 4.1. *$(T, u)_{\{v\}}$ is the subtree rooted at v not containing u, as illustrated in figure 15.*

Notation 4.2. *$(T, u)_{\{1\}}$ is the subtree rooted at a child of u containing 1 and not containing u, as illustrated in the sae figure 15.*

Notation 4.3. *$(T, u)_{\setminus L}$ is the subtree rooted at u not containing any of the first L vertices, as illustrated in figure 16.*

We conjecture two observations:

Conjecture 4.1

Let $G(n)$ be a tree graph grown under a preferential attachment model starting with a singleton and with an increasing label starting from 1. Let $u > 1$ be a vertex not 1. Let $N(u)$ be the set of vertices neighboring u.

Then in the limit as n goes to infinity, the expected value of the size of $(G, u)_{\{1\}}$ is pairwise bounded from below by the size of $(G, u)_{\{v\}}$ for all v in $N(u)$.

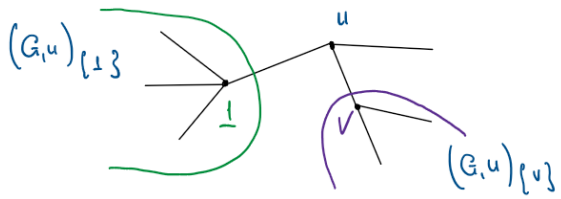

Figure 15: Examples of how conjecture 1 applies.

Conjecture 4.1 asserts that as $n \to \infty$, for any vertex $u > 1$, we can expect that computing $\varphi(u)$ reduces to computing the size of the component containing vertex 1 once u is removed.

Conjecture 4.2

Let $G(n)$ be a tree graph grown under a preferential attachment model starting with a singleton and with an increasing label starting from 1. Suppose the first L vertices are known. Let $N([L])$ be the set of vertices neighboring any of those L vertices. Let u be the $L+1^{st}$ vertex.

Then in the limit as n goes to infinity, the expected value of the size of $(G,u)_{\setminus[L]}$ is pairwise bounded from below by the size of $(G,v)_{\setminus[L]}$ for all v in $N([L])$.

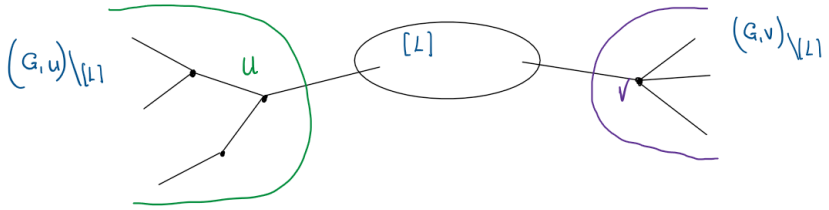

Figure 16: Examples of how conjecture 2 applies.

Conjecture 4.2 asserts that for any vertices u and v neighboring the first L vertices, as $n \to \infty$, if u appears before v, then we can expect that if all L vertices are removed, the component containing u will be as least as big as that containing v.

Assuming the two conjectures, we combine them to give support to the proposed algorithm 3: suppose in returning the first L vertices, we have returned the set $\mathcal{X}_L$, we now look at the neighbors of $\mathcal{X}_L$:

1. Conjecture 4.1 implies that for any vertex $u \in N(\mathcal{X}_L)$, it is reasonable to expect that $\varphi(u)$ reduces to the size of the component containing $\mathcal{X}_L$, once u is removed.

2. Continue on from point (1), we observe that for any u and $v \in N(\mathcal{X}_L)$, comparing $\varphi(u)$ and $\varphi(v)$ reduces to comparing the sizes of $(G,u)_{\setminus[L]}$ and $(G,v)_{\setminus[L]}$, as seen in figure 16, since the portions containing $\mathcal{X}_L$ and what grow out of $\mathcal{X}_L$ are the same.

3. Conjecture 4.2 implies that in comparing $(G,u)_{\setminus[L]}$ and $(G,v)_{\setminus[L]}$, if u appeared before v, then it is reasonable to expect that $|(G,u)_{\setminus[L]}| \geq |(G,u)_{\setminus[L]}|$, which implies that $\varphi(u) \leq \varphi(v)$.

4. Hence, in doing pairwise comparison of the φ values, we could expect that vertex $L+1$ will have the minimum φ value.

4.3 Simulation

4.3.1 Simulation Data

We would like to study the two algorithms proposed. We will test the algorithms on the same data set used in section 4.3.1..

4.3.2 Simulation Results

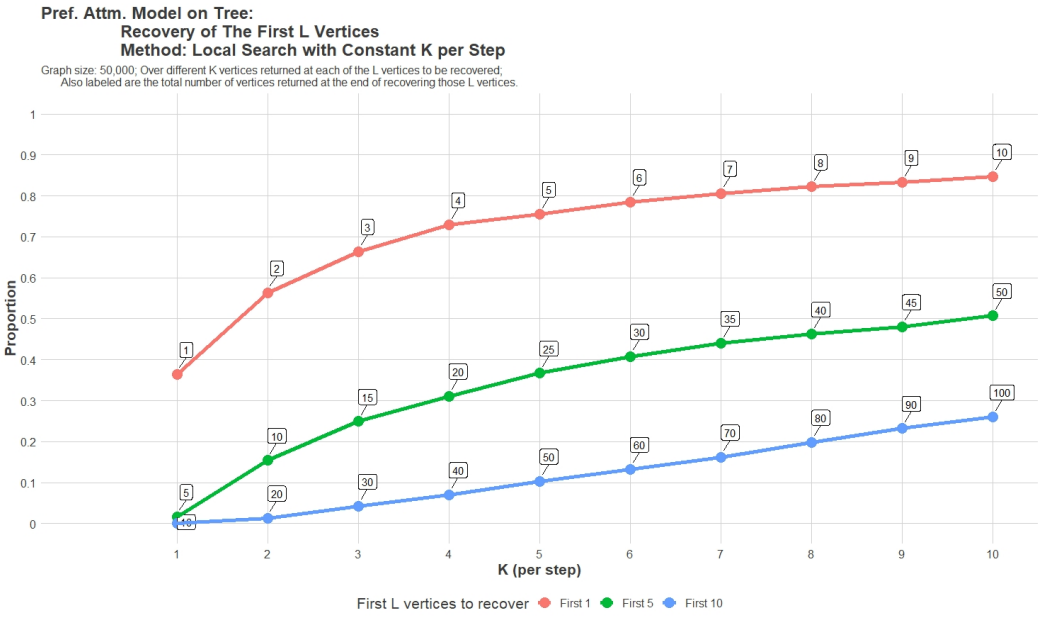

Figure 17: Recovery rates of the first L vertices on *preferential attachment* model on tree via the *sequential local search* across different K from 1 to 10 at each step on graphs of size $50,000$. Also labeled is the total number of vertices returned at the end of the search.

Figures 17 and 18 show the recovery rate of the first $1, 5$ and 10 vertices under the *local search* algorithm on graphs of size $50,000$. At each step, we returned K from 1 to 10 vertices, and larger K such as 10 to 50 per step. We have some observations:

1. The recovery rates decreased significantly when we wanted to recover 5 and 10 vertices, compared to simply recovering vertex 1.

2. Although the recovery rates appear low, they appear to be on an increasing trend, as seen in figure 18, where it showed that it was possible to achieve accuracy of over 50% if we were willing to return a total of 50 or 400 vertices (for recovering the first 5 and 10 vertices respectively).

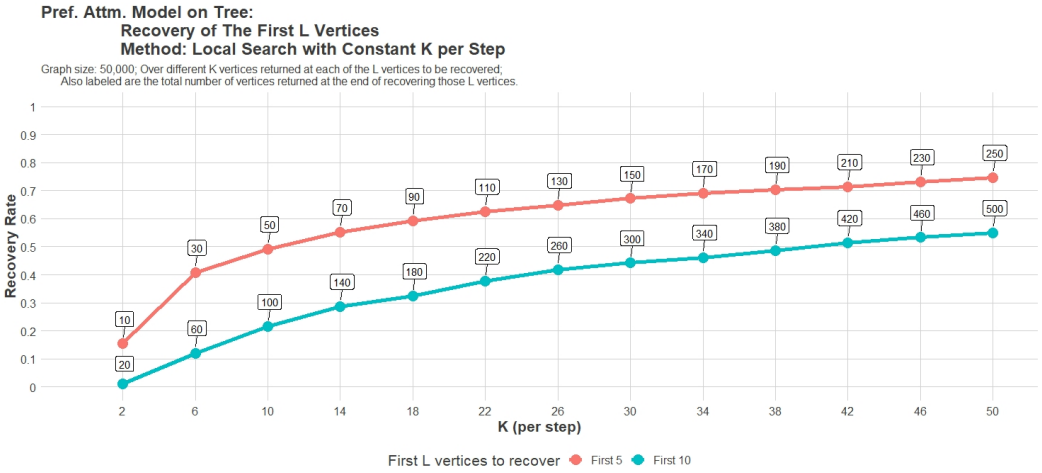

Figure 18: Recovery rates of the first L vertices on *preferential attachment* model on tree via the *sequential local search* across different K from 2 to 50 at each step on graphs of size $50,000$. Also labeled is the total number of vertices returned at the end of the search.

Figure 19 shows the recovery rate of the first 5 and 10 vertices under the *local search* algorithm across different graph sizes. At each step, we returned K being $10, 30$ and 50 vertices. We have some observations:

1. Fixing the number of vertices returned per step (K), the recovery rates remained essentially unchanged over a wide range of graph sizes whether we returned $10, 30$ or 50 vertices at each step.

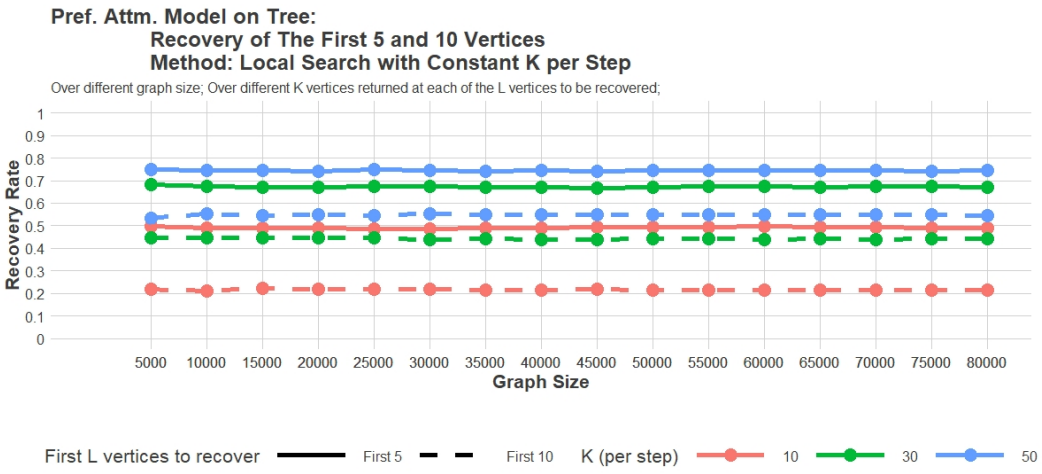

Figure 19: Recovery rates of the first 5 and 10 vertices on *preferential attachment* model on tree via the *sequential local search* with $K \in \{10, 30, 50\}$ across different graph sizes.

We note the significant observation of the recovery rates being stable across a wide range of graph sizes. This gives an impression that we would only need to tune K, the number of vertices returned at each step, to increase the recovery rate. This behavior is further confirmed by figures 20 and 21.

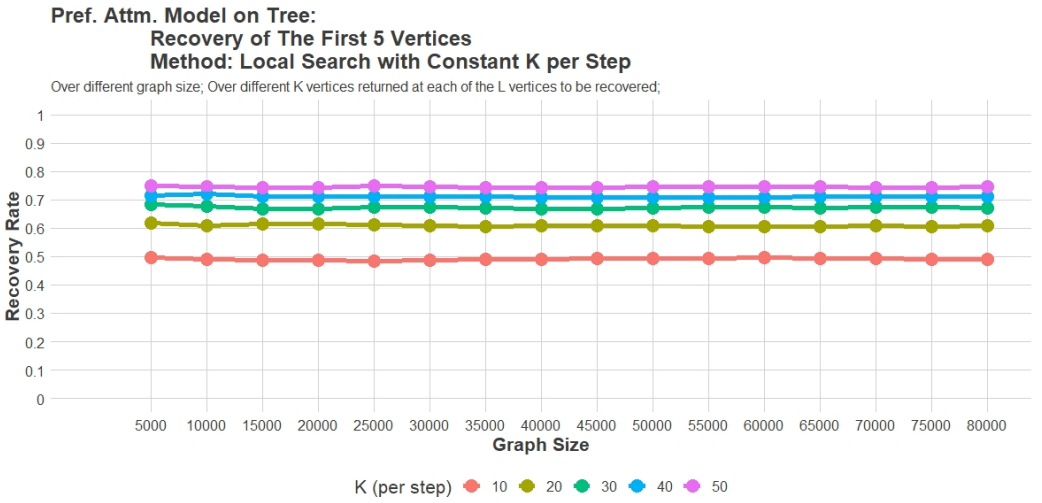

Figure 20: Recovery rates of the first 5 vertices on *preferential attachment* model on tree via the *sequential local search* across different K.

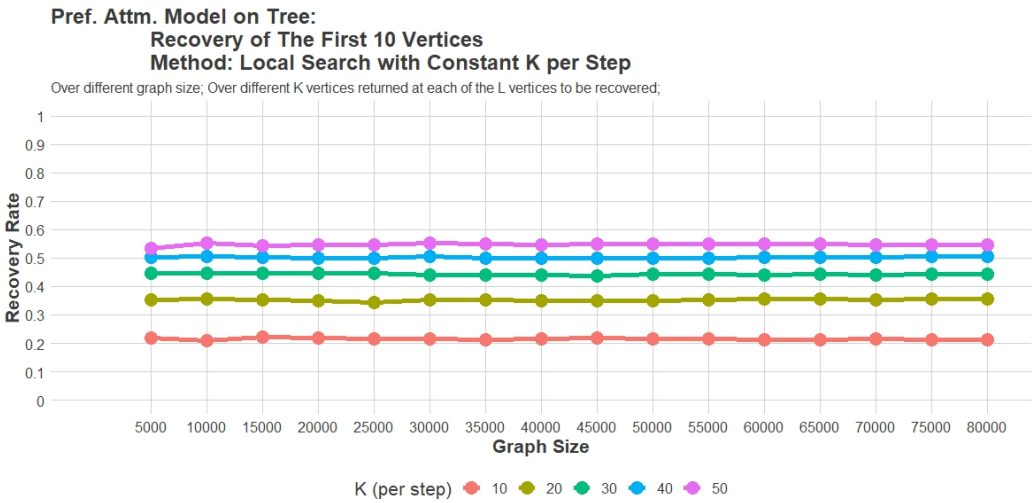

Figure 21: Recovery rates of the first 10 vertices on *preferential attachment* model on tree via the *sequential local search* across different K.

Given those observations, we can now compare the *local search* algorithm with the simple search of simply returning K vertices with the smallest φ values.

Figure 22 shows the recovery rates between the two algorithms. We have some observations:

1. There appear no significant differences between the two algorithms: for the same number of vertices returned, the accuracy of a *simple search* closely resembles that of the more complicated *sequential search*.

2. However, when returning more vertices, the gap in the accuracy appear to be widening, as illustrated by the gaps on the far right of each set of curves.

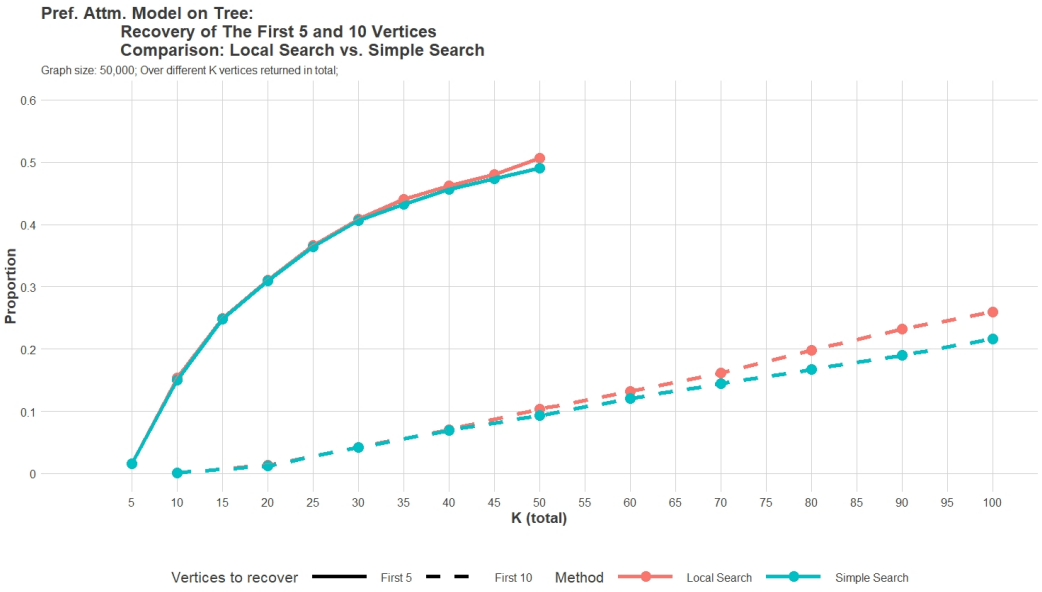

Figure 22: Comparison of the recovery rates of the first 5 and 10 vertices via the *local search* and the *simple search* algorithms.

Those observations give us an impression that while there is little difference between the two algorithms initially, the gap would start to grow as we return more vertices. We plan to further study the trade-off between accuracy and precision of returning the first L vertices.

4.4 A Second Look at the Problem

We note the significant drop in the accuracy rates when we started recovering more than just vertex 1. We would like to look further into this behavior.

We will use a variant of the *local search* algorithm as described in 3. In particular, we will look to recover vertex L assuming that we know exactly all the $\mathcal{X} = \{1, 2, \ldots, L-1\}$ vertices and will search among the neighbors of $\mathcal{X}$. We will refer to this as *local search with perfect knowledge*. The objective is to study how hard the problem becomes once we start looking for vertices away from 1.

Figures 23, 24 and 25 show the recovery rates of a local search with perfect knowledge, using the methods of both the φ values and degree. We have some observations:

1. Comparing between recovering vertex 2 and vertex 10, the drops in accuracy can be significant.
2. We could expect the accuracy of using the maximum degree to closely match that of using the φ values for the earliest vertices. However, as we move further away from vertex 1, the difference in accuracy can get more significant, as illustrated in figure 25.

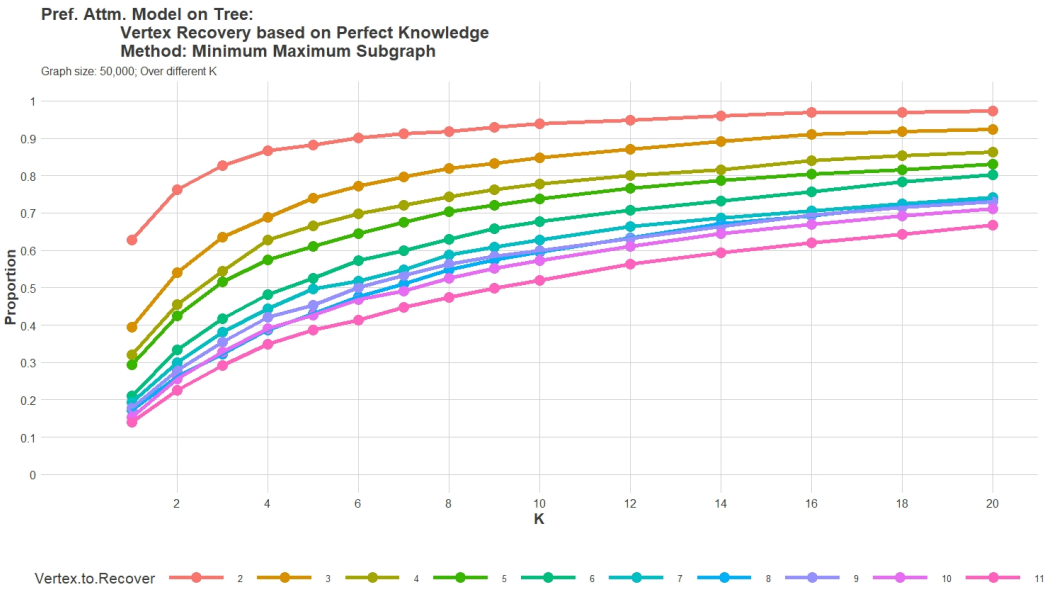

Figure 23: Recovery rates of the first 11 vertices via a local search with perfect knowledge using φ values.

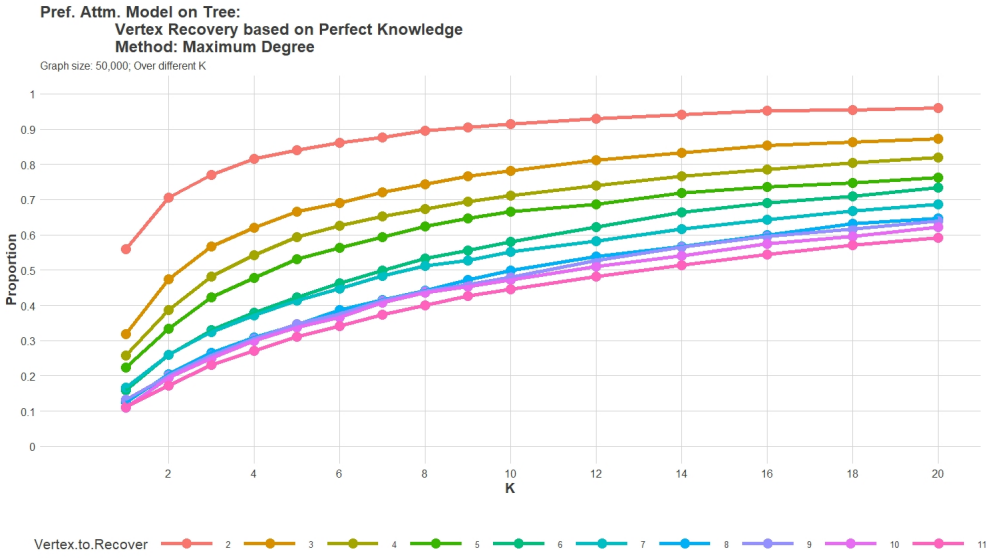

Figure 24: Recovery rates of the first 11 vertices via a local search with perfect knowledge using vertex degree.

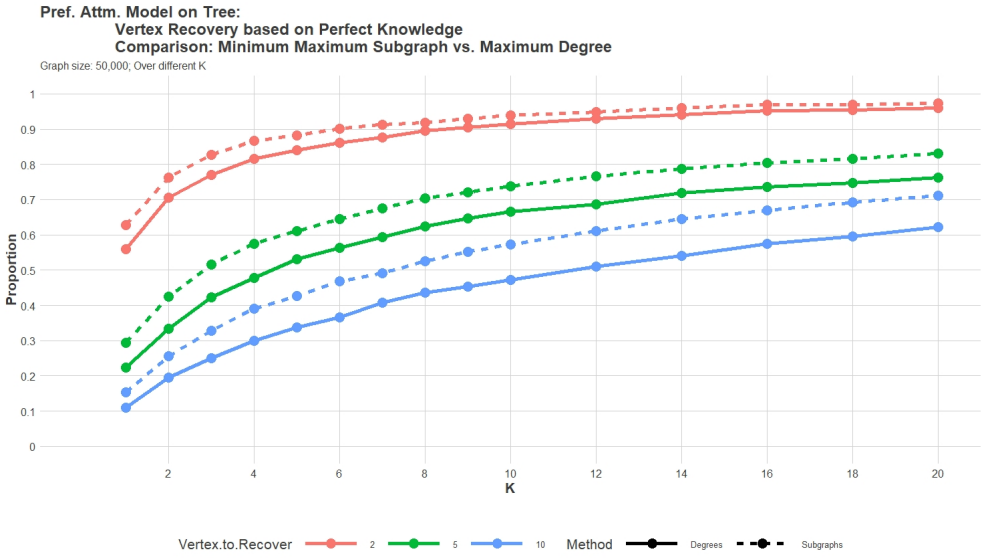

Figure 25: Comparison of the recovery rates via a local search with perfect knowledge.

We observe that the problem of recovering vertices after 1 can get hard quickly. Although the experiment appears to show that it would still be possible to achieve certain high level of accuracy, we will have to trade a lot in terms of being concise. In particular, we will have to return a much larger range of vertices to get to certain accuracy.

This observation is indeed noted by Lugosi and Pereira in [10]. The paper studies the same problem of recovering the seed graph. Although the paper focuses only *uniform attachment* model on tree, we suspect that the result will carry to a large extent over to that of *preferential attachment* model on tree. The paper shows that even when we assume additional structures on the seed graph, as soon as the seed graph exceed certain size, any algorithm must miss at least half the number of vertices with probability at least some ε. We cite the results here.

Theorem 4.1: Uniform attachment model on tree, seed graph is a path, [10]

Let $\varepsilon \in (0, e^{-e^2})$. Suppose that T_n is a uniform attachment tree with the seed graph, S_l, being a line graph of size l for

$$l \leq \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}$$

then for any $n \geq 2l$ (where n is the graph size), any seed-finding algorithm that outputs a vertex set H of size l has

$$\mathbb{P}\left(\left|H \cap P_l\right| \leq \frac{l}{2}\right) \geq \varepsilon$$

Theorem 4.2: Uniform attachment model on tree, seed graph is a star, [10]

Let $\varepsilon \in (0, e^{-e^2})$. Suppose that T_n is a uniform attachment tree with the seed graph, S_l, being a star graph of size l for

$$l \leq \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}$$

then for any $n \geq 2l$ (where n is the graph size), any seed-finding algorithm that outputs a vertex set H of size l has

$$\mathbb{P}\left(\left|H \cap P_l\right| \leq \frac{l}{2}\right) \geq \varepsilon$$

5 Conclusion

In this paper, we studied the problem of recovering the first vertices in a model assumed to have grown under the *preferential attachment* rules. We focused on a special case where the graphs are trees. We revisited the approach of looking at the maximum subgraph created when removing each vertex. We provided a more detailed proof and studied the efficiency with simulations. We summarize the key findings below:

1. In recovering vertex 1, we can get accurate in terms of finding a set of vertices in which 1 belong and that the set is small very quickly. In particular, we can achieve an accuracy of about 90% by returning only 20 vertices regardless of the graph size, as seen in figure 7.

2. Once we want to recover more than just vertex 1, the problem becomes much harder quickly. Not only is the drop in accuracy significant, but we would now need to return a lot more vertices to be of certain confidence that the first L vertices have been fully recovered.

3. This phenomenon brings to our attention the trade-off between being

   $$\underbrace{\textit{accurate}}_{\substack{\text{that } L \text{ vertices} \\ \text{have been recovered}}} \quad \textit{vs.} \quad \underbrace{\textit{precise}}_{\substack{\text{that only a small number} \\ \text{of vertices have been returned}}}$$

   The findings from section 4.3.2 suggest that if we are willing to forgo being precise, we could improve the accuracy, although the trade-off appear significant.

4. Once we have fixed the objective (number of vertices to recover), the method (either using vertex degree or the φ values), and the approach (local search of simple search), the accuracy remains constant regardless of graph sizes, ranging from size of $5,000$ to even 15 times larger of size $75,000$. This suggests that the difficulty of the problem lies in the structure of such a model and not so much on the size.

5. Although always trailing behind in terms of accuracy, simply looking at the vertex degree can get us quite far, compared to using the more complicated algorithm of looking at the φ values. However, computing vertex degree is straightforward and is much cheaper. As such, when the computation cost can get expensive, we propose a two-step approach where (1) we look at the vertex degree to narrow down the set of suspected vertices, and then (2) compute the φ values with little compromise on the accuracy, as seen in section 3.4.

6 References

[1] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[2] B´ela Bollobás, Oliver Riordan, Joel Spencer, and Gábor Tusnády. The degree sequence of a scale-free random graph process. *Random Structures & Algorithms*, 18(3):279–290, 2001.

[3] Sébastien Bubeck, Luc Devroye, and Gábor Lugosi. Finding adam in random growing trees. *Random Structures Algorithms*, 50(2), 2016.

[4] Fan Chung and Linyuan Lu. *Complex Graphs and Networks*. Number 107. American Mathematical Society, 2006. CBMS Regional Conference Series in Mathematics.

[5] S. N. Dorogovtsev, J. F. F. Mendes, and A. N. Samukhin. Structure of growing networks with preferential linking. *Physical Review Letters*, 85(21), 2000.

[6] Alan Frieze and Wesley Pegden. Looking for vertex number one. *Ann. Appl. Probab.*, 27(1):582–630, 02 2017.

[7] Bela A Frigyik, Amol Kapila, and Maya R Gupta. Introduction to the dirichlet distribution and related processes. *Department of Electrical Engineering, University of Washignton, UWEETR-2010-0006*, (0006):1–27, 2010.

[8] Remco van der Hofstad. *Random Graphs and Complex Networks*, volume 1 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, 2016.

[9] Svante Janson. Limit theorems for triangular urn schemes. *Probability Theory and Related Fields*, 134(3):417–452, 2006.

[10] Gábor Lugosi and Alan S. Pereira. Finding the seed of uniform attachment trees. *Electron. J. Probab.*, 24:15 pp., 2019.

[11] Mark Newman. *Networks*. Oxford university press, 2018.

[12] Derek De Solla Price. A general theory of bibliometric and other cumulative advantage processes. *Journal of the American Society for Information Science*, 27(5):292–306.

[13] Sheldon Ross. *A first course in probability*. Pearson, 2014.