The Network Completion Problem: Inferring Missing Nodes and Edges in Networks

Myunghwan Kim* Jure Leskovec†

Abstract

While the social and information networks have become ubiquitous, the challenge of collecting complete network data still persists. Many times the collected network data is incomplete with nodes and edges missing. Commonly, only a part of the network can be observed and we would like to infer the unobserved part of the network. We address this issue by studying the *Network Completion Problem*: Given a network with missing nodes and edges, can we complete the missing part?

We cast the problem in the Expectation Maximization (EM) framework where we use the observed part of the network to fit a model of network structure, and then we estimate the missing part of the network using the model, re-estimate the parameters and so on. We combine the EM with the Kronecker graphs model and design a scalable Metropolized Gibbs sampling approach that allows for the estimation of the model parameters as well as the inference about missing nodes and edges of the network.

Experiments on synthetic and several real-world networks show that our approach can effectively recover the network even when about half of the nodes in the network are missing. Our algorithm outperforms not only classical link-prediction approaches but also the state of the art Stochastic block modeling approach. Furthermore, our algorithm easily scales to networks with tens of thousands of nodes.

1 Introduction

Network structures, such as social networks, web graphs and networks from systems biology, play important roles in many areas of science and our everyday lives. In order to study the networks one needs to first collect reliable large scale network data [23]. Even though the challenge of collecting such network data somewhat diminished with the emergence of Web, social media and online social networking sites, there are still many scenarios where collected network data is not completely mapped. Many times the collected network data is incomplete with nodes and edges missing [15]. For example, networks arising from the popular social networking services are not completely mapped due to network boundary effects, *i.e.*, there are people who do not use the social network service and so we cannot observe their connections. Similarly, when a social scientist collects the data by surveying people, he or she may not have access to certain parts of the network or people may not respond to the survey. For example, complete social networks of hidden or hard-to-reach populations, like drug injection users or sex workers and their clients, are practically impossible to collect. Thus, network data is often incomplete, which means that some nodes and edges are missing from the dataset. Analyzing such incomplete network data can significantly alter our estimates of network-level statistics and inferences about structural properties of the underlying network [15].

We systematically investigate the *Network Completion Problem* where we aim to infer the unobserved part of the network. While the problem of missing link inference [6, 10] has been thoroughly studied before, we consider a problem where both edges and nodes are missing. Thus, the question that we address here is: given an incomplete network with missing nodes and edges, can we fill in the missing parts? For example, can we estimate the structure of the full Facebook social network by only observing/collecting a part of it? Figure 1(a) illustrates the task. There is a complete network represented by adjacency matrix H and some nodes and corresponding edges are missing from it. We only observe network (matrix) G, the non-missing part of H, and aim to infer the missing nodes and edges, *i.e.*, the missing part Z.

The problem that we aim to solve here is particularly challenging in a sense that we only assume the partial knowledge of the network structure and based on that we aim to predict the structure of the unobserved part of the network. In particular, we work only with the network structure itself, with no additional data about the properties (attributes or features) of the nodes and edges of the network. This means that we aim to infer the missing part of the network solely based on the connectivity patterns in the observed part.

*Stanford University, CA 94305. mykim@stanford.edu

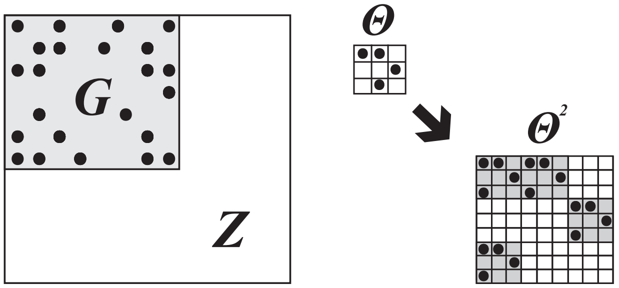
†Stanford University, CA 94305. jure@cs.stanford.edu

Present work: A model-based approach. In order to capture the connectivity patterns in the observed part of the network and use this knowledge to complete the unobserved part of the network, one inherently requires a model of the structure of real-networks. While many network models have been proposed [6, 1, 16, 20], our requirements here are somewhat specific. First, since we aim to work with large scale network data, the model and the parameter estimation procedure needs to be scalable. Second, it is desirable for the network model to be statistical in nature so that we can efficiently model the probability distribution over the missing part of the network.

The Kronecker graphs model [18] satisfies above requirements. The model is provably able to simultaneously capture several properties of network structure, like heavy-trailed degrees, small diameter and local clustering of the edges [18, 22]. Moreover, Kronecker graphs model parameters can be efficiently estimated. In practice, the model reliably captures the structure of many large real-world networks [17].

Based on the Kronecker graphs model, we naturally cast the problem of network completion into the Expectation Maximization (EM) framework where we aim to estimate the Kronecker graphs model parameters as well as the edges in the missing part of the network. In this EM framework, we develop the KronEM algorithm that alternates between the following two stages. First, we use the observed part of the network to estimate the parameters of the network model. The estimated model then gives us a way to infer the missing part of the network. Now, we act as if the complete network is visible and we re-estimate the model. This in turn gives us a better way to infer the missing part of the network. We iterate between the model estimation step (the M-step) and the inference of the hidden part of the network (the E-step) until the model parameters converge.

However, in order to make this approach feasible in practice, there are several challenges that we need to overcome. In particular, the model parameter fitting as well as the estimation of missing edges are both computationally intractable. Moreover, the parameter search space is super-exponential in the number of nodes N in complete network H as three sets of parameters/variables need to be estimated: the Kronecker graph parameters (usually just 4 parameters), the mapping of the nodes of the Kronecker graph to those of the observed and missing part of the network (N! possible mappings), and the inference of the missing edges of the network ($O(2^N)$ configurations). However, we develop a set of scalable techniques that allow for fast and efficient model parameter estimation as well as the inference of missing nodes and edges of the network.



(a) The Network Completion Problem

(b) Kronecker Graphs model

Figure 1: (a) The network completion problem: There is a complete graph adjacency matrix H. We only observe part G of H, and aim to infer the missing part Z. (b) Kronecker Graphs model: Parameter matrix Θ and the second Kronecker power Θ^2. Notice the recursive structure of matrix Θ^2, where entries of Θ are recursively expanded with miniature copies of Θ itself.

Experiments on synthetic and large real-world networks show that our approach reliably predicts the individual missing edges as well as effectively recovers the global network structure, even when about half of the nodes of the network are missing. In terms of performance, our algorithm compares favorably to classical link-prediction approaches and the state of the art Stochastic block modeling [10] approach.

KronEM has several advantages. In contrast to the Stochastic block model, KronEM requires a small number of parameters and thus does not overfit the network. It infers not only the model parameters but also the mapping between the nodes of the true and the estimated networks. The approach can be directly applied to cases when collected network data is incomplete. Overall, KronEM provides an accurate probabilistic prior over the missing network structure. Furthermore, KronEM easily scales to large networks.

2 Background and Related Work

Next, we survey the related work and then introduce the Kronecker graphs model. The network completion has been studied in the context of survey sampling [11] and the inference of missing links mainly in social and biological networks [10, 6]. Related to our work is also the problem of link prediction, which is usually formulated in the following way: given a network up to time t, predict what new edges will occur in the future. Link prediction has been extensively studied in social and information networks [21, 28] as well as biological networks [8]. Similarly, the network reconstruction problem that has been mainly studied in the context of biological networks, aims to infer the missing parts of networks [31, 3]. Another related setting is the network inference problem where a virus diffuses through the network and we only observe node infection times (but not who infected whom). The goal then is to infer hidden underlying network [26, 9]. In contrast, the network inference and network reconstruction assume that identities and properties (features) of missing nodes (proteins) are known and one aims to infer their links. Overall, these previous works assume that the nodes are already present in the network, we consider a different setting where both nodes and edges are missing.

Our work also relates to the matrix completion problem [4, 12] where a data matrix with missing entries is given and the aim is to fill the entries. However, it also differs in a sense that real-world networks have very different structure and properties (e.g., heavy-tailed degree distributions) than the real-valued data matrices usually considered in the matrix completion problem.

Kronecker graphs model. Next we briefly introduce the Kronecker graphs model of real-world networks [18] that we later adopt to develop a method for inferring missing nodes and edges. To introduce the Kronecker graphs model, we first define the Kronecker product of two matrices. For $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{m' \times n'}$ Kronecker product $A \otimes B \in \mathbb{R}^{mm' \times nn'}$ is defined as:

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}$$

And Kronecker product of two graphs is the Kronecker product of their adjacency matrices.

The Kronecker graphs model is then defined by a small Kronecker parameter matrix $\Theta \in [0,1]^{N_0 \times N_0}$ where every entry of the matrix Θ can be interpreted as a probability. We can Kronecker power Θ to obtain larger and larger stochastic graph adjacency matrices. The resulting matrices are naturally self-similar as the pattern of subgraph resembles that of the entire graph as illustrated by Figure 1(b). For example, after k powerings of Θ we obtain $\Theta^k \in [0,1]^{N_0^k \times N_0^k}$. Now Θ^k defines a probability distribution over graphs on N_0^k nodes where each entry $(\Theta^k)_{ij}$ of this large Θ^k can be interpreted as a probability that edge (i,j) exists. Thus, to obtain a realization K of a Kronecker graph defined by parameter matrix Θ, we include edge (i,j) in K according to its probability of occurrence $(\Theta^k)_{ij}$.

Kronecker graphs model is particularly appropriate for our task here, since it has been shown that it can reliably model the structure of many large real-world networks [17, 19]. Our task now is to extend the model to networks with missing data and to develop a procedure that will at the same time infer model parameters as well as the missing part of the network.

3 The Network Completion Problem

We begin by introducing the problem of inferring missing nodes and edges in networks. We then propose KronEM, an EM algorithm based on Kronecker graphs model to solve the network completion problem.

3.1 Problem Definition. We consider that there is a large complete (directed or undirected) network $H(N, E)$ which we cannot completely observe as the information about some nodes is missing, *i.e.*, we do not know the edges of these missing nodes. In contrast, we are able to fully observe the induced subgraph $G(N_G, E_G)$ of H. G is the observed part of the network H. Now the task is to infer the missing part Z of the network H. We denote the missing nodes and edges in Z as N_Z and E_Z, respectively. E_Z are therefore edges with at least one endpoint in N_Z, *i.e.*, edges between pairs of nodes in N_Z and those with one endpoint in N_Z and the other in N_G. Moreover, we assume that the amount of missing data (*i.e.*, size of N_Z) is known. If not, then standard methods for estimating the size of hidden or missing populations [24, 13] can be applied.

Considering H and G as the adjacency matrices, the problem is equivalent to a variant of matrix completion problem as illustrated in Figure 1(a). We have a binary matrix H of which we only observe block G and the task is to infer the missing part Z. Our goal is to determine whether each entry in Z should be set to 0 or 1. In contrast to the classical matrix completion or missing-link inference problem where the assumption is that random entries of H are missing (and so low-rank matrix approximation methods can be thus applied), in our case complete rows and columns of H are missing, which makes the problem much more challenging.

3.2 Proposed Approach: Kronecker EM. We model the complete network H with a Kronecker graphs model. Then, the observed part G and the missing part Z are linked probabilistically through the model parameters Θ:

$$Z \sim P_H(Z|G, \Theta)$$

The objective of the network completion problem is to find the most likely configuration of the missing part Z. However, we first need to estimate the parameters Θ to establish a link between the observed G and unobserved Z. This naturally suggests the Expectation-Maximization (EM) approach, where we consider the missing edges in Z as latent variables. Therefore, we iteratively update Θ by maximizing the likelihood $P(G, Z|\Theta)$ and then update Z via $P(Z|G, \Theta)$.

In addition to the missing part Z, one more set of variables is essential for the EM approach to work. We have to distinguish which nodes of the inferred

Kronecker graph belong to part G and which to part Z. We think of this as a mapping σ between the nodes of the inferred network $\hat{H}$ and the observed part G. Each node in H and in $\hat{H}$ has unique id $1, ..., N$. Then the mapping σ is a permutation of the set $\{1, \ldots, N\}$ where the first N_G elements of σ map the nodes of $\hat{H}$ to the nodes of G and the remaining N_Z elements of σ map the nodes of Z. Under the Kronecker graphs model, we define the likelihood $P(G, Z, \sigma|\Theta)$:

(3.1)

$$P(G, Z, \sigma|\Theta) = \prod_{u \sim v} [\Theta^k]_{\sigma(u)\sigma(v)} \prod_{u \not\sim v} \left(1 - [\Theta^k]_{\sigma(u)\sigma(v)}\right)$$

where Θ^k is the stochastic adjacent matrix generated by the Kronecker parameter matrix Θ, σ maps the nodes, and $u \sim v$ represents an edge in $H = G \cup Z$.

This means that in order to compute $P(G|\Theta)$ we integrate $P(G, Z, \sigma|\Theta)$ over two sets of latent variables: the edges in the missing part Z, and the mapping σ between the nodes of G and the nodes of the inferred network $\hat{H}$. Using the EM framework, we develop KronEM algorithm where in the E-step we consider the joint posterior distribution of Z and σ by fixing Θ and in the M-step we find optimal Θ given Z and σ:

E-step:

$$(Z^{(t)}, \sigma^{(t)}) \sim P(Z, \sigma|G, \Theta^{(t)})$$

M-step:

$$\Theta^{(t+1)} = \arg \max_{\Theta \in (0,1)^2} \quad \mathbb{E}\left[P(G, Z^{(t)}, \sigma^{(t)}|\Theta)\right]$$

Note that the E-step can be viewed as a procedure that completes the missing part of the network.

Now the issue is the E-step is that there is no closed form for the posterior distribution $P(Z, \sigma|G, \Theta)$. The basic EM algorithm is also problematic because we cannot compute the expectation in the M-step. To solve this, we introduce the Monte-Carlo EM algorithm (MCEM) that samples Z and σ from the posterior distribution $P(Z, \sigma|G, \Theta)$ and averages $P(G, Z, \sigma|\Theta)$ over these samples [30]. This MCEM algorithm has an additional advantage: if we take the Z and σ that maximize $P(G, Z, \sigma|\Theta)$, they are the most likely instances of the missing part and the node mapping. Thus, no additional algorithm for the completion of the missing part is needed.

However, we still have several challenges to overcome. For example, sampling from the posterior distribution $P(Z, \sigma|G, \Theta)$ in E-step is nontrivial, since no closed form exists. Furthermore, since the sample space over Z and σ is *super-exponential* in the network size ($O(2^{NN_Z})$ for Z and $N!$ for σ), which is unfeasible for the size of the networks that we aim to work with here.

Algorithm 1 E-step of KronEM

```
input: Observed network $G$, Kronecker parameter $\Theta$,
        # of warm-up iterations $W$, # of samples $S$
output: Set of samples of $Z$ and $\sigma$
Initialize $Z^{(0)}, \sigma^{(0)}$
for $i = 1$ to $(W + S)$ do
    $Z^{(i)} \leftarrow P(Z|G, \sigma^{i-1}, \Theta)$    // Gibbs sampling of $Z$
    $\sigma^{(i)} \leftarrow P(\sigma|G \cup Z^{(i)}, \Theta)$    // Gibbs sampling of $\sigma$
end for
$\mathbb{Z} \leftarrow (Z^{(W+1)}, \cdots, Z^{(W+S)})$
$\Sigma \leftarrow (\sigma^{(W+1)}, \cdots, \sigma^{(W+S)})$
return $\mathbb{Z}, \Sigma$
```

Focusing on these computational challenges, we develop E-step and M-step in the following sections.

3.3 E-step. As described, it is difficult to sample directly from $P(Z, \sigma|G, \Theta)$, i.e., it is not clear how to at the same time sample an instance of Z and σ from $P(Z, \sigma|G, \Theta)$.

However, we make the following observation. When either Z or σ is fixed, we can sample the other (σ or Z). If the node mapping σ is fixed, then the distribution of edges in Z is determined by Θ and σ. In detail, we first generate a stochastic adjacency matrix Θ^k by using Kronecker parameter matrix Θ. Then, since the node mapping σ tells us which entries of Θ^k belong to the missing part Z, the edges in Z are obtained by a series of Bernoulli coin-tosses with probabilities defined by the entries of Θ^k and the node mapping σ. Conversely, when the missing part Z is fixed so that the complete network H is given, sampling a node mapping $\sigma \sim P(\sigma|\Theta, G, Z)$ conditioned on Θ and H (*i.e.*, $H = G \cup Z$) is tractable as we show later.

By these observations, we generate many samples of $P(Z, \sigma|G, \Theta)$ by using Gibbs sampling, where we fix σ and sample new Z, and then fix Z and sample new σ, and so on. Algorithm 1 gives the E-step using the Gibbs sampling approach that we now describe in detail.

Sampling $\mathbf{Z}$ from $\mathbf{P(Z|G, \sigma, \Theta)}$. Given a node mapping σ and a Kronecker parameter matrix Θ, sampling the missing part Z determines whether every edge Z_{uv} in Z is present or not. We model this with Bernoulli distribution with parameter $[\Theta^k]_{\sigma(u)\sigma(v)}$ where $\sigma(u)$ represents mapping of node u in G to a row/column of the stochastic adjacent matrix Θ^k. Entry $[\Theta^k]_{uv}$ of matrix Θ^k can be thus interpreted as the probability of an edge (u, v). However, the number of possible edges in Z is quadratic $O(N^2)$, so it is infeasible to sample Z by flipping $O(N^2)$ coins each with probability $[\Theta^k]_{uv}$. Thus, we need a method that infers the missing part Z in more efficiently than in quadratic time.

We notice that an individual edge (u, v) in a Kronecker graph can be sampled as follows. First we sample two N_0-ary (i.e., binary) vectors u and v of length k with entries u_l and v_l $(l = 1, \ldots, k)$. The probability of (u_l, v_l) taking value (i, j) (where $i, j \in \{1, \ldots, N_0\}$) is proportional to the corresponding entry Θ_{ij} of the Kronecker parameter matrix. Thus, pairs (u_l, v_l) act as selectors of entries of Θ. Given two such vectors, we compute the endpoints u, v of edge (u, v) [17]:

$$(u, v) = (\sum_{l=1}^{k} u_l N_0^{l-1}, \sum_{l=1}^{k} v_l N_0^{l-1})$$

where N_0 is the size of Kronecker parameter matrix Θ (usually $N_0 = 2$) and $k = \log_{N_0} N$. From this procedure we obtain an edge from a full Kronecker graph $\hat{H}$. Since we are interested only in the missing part Z of $\hat{H}$, we accept an edge (u, v) only if at least one of u and v is a missing node. We iterate this process until we obtain E_Z edges. Graphically, if the sampled edge (u, v) belongs to part Z in Figure 1(a), we accept it; otherwise, we reject it and sample a new edge. As the rejection rate is approximately $E_G/(E_Z + E_G)$, the expected running time to obtain a single sample of the missing part Z is $O(E_G)$ (down from the original $O(N^2)$).

Even though we reduced the time to obtain a single sample of Z, this is still prohibitively expensive given that we aim to obtain millions of samples of Z. More precisely, to generate S samples of Z, the above procedure takes $O(SE_G)$ time. Generally, we need more samples S as the size of missing part E_Z increases. Thus, the above procedure will be too slow.

We started with $O(N^2)$ algorithm for sampling Z and brought it down to linear time $O(E_G)$. We further improve this by making sampling a single Z effectively in a constant-time operation. We develop a Metropolis-Hastings sampling procedure where we introduce a Markov chain. We do this by making samples dependent, *i.e.*, given sampled $Z^{(i)}$, we generate the next candidate sample $Z^{(i+1)}$. To obtain $Z^{(i+1)}$ we consider a proposal mechanism which takes $Z^{(i)}$, removes one edge randomly from it, and adds another one to it. We then accept or reject this new $Z^{(i+1)}$ based on the ratio of likelihoods. In this way, we reduce the complexity of obtaining $Z^{(i+1)}$ to $O(1)$, since all we need to do is to take a previous sample $Z^{(i)}$, delete an edge and add a new one. Therefore, it takes only $O(S)$ time to acquire S samples of Z. We give further details of the Metropolis-Hastings sampling procedure with the detailed balance correction factor in the extended version of the paper [14].

Sampling σ from $\mathbf{P}(\sigma|\mathbf{G}, \mathbf{Z}, \mathbf{\Theta})$. The second part of the Gibbs sampling obtains the samples of node mapping σ. The complete network H (*i.e.*, $H = G \cup Z$) and the Kronecker parameters Θ are given and we want to sample the node mapping σ Here we also have an issue because the sample space of all node mappings σ (i.e., all permutations) is $N!$. We develop a Metropolis sampling method that quickly converges to good node mappings. The sampling proposal mechanism selects two random elements of σ and swaps them. By using such proposal mechanism, σ changes only locally and thus the computations are very fast. However, the probability of acceptance of the transition $\sigma \rightarrow \sigma'$ (swapping random positions i and j) needs to be derived as the detailed balance conditions are not simply met. We provide the details in the extended version [14].

Algorithm 2 M-step of KronEM

```
input: Observed network $G$
       Gibbs samples $\mathbb{Z}$ and $\Sigma$ obtained in E-step
       # of gradient ascent updates $D$
output: New Kronecker parameter matrix $\Theta$
initialize $\Theta$, set $L \leftarrow \frac{1}{D}$ length($\mathbb{Z}$)
for $j = 1$ to $D$ do
  for $i = 1$ to $L$ do
    $(Z, \sigma) \leftarrow (\mathbb{Z}_{i+(j-1)L}, \Sigma_{i+(j-1)L})$
    $\Delta_i^{(j)} \leftarrow \frac{\partial}{\partial\Theta} P(G, Z, \sigma|\Theta)$   // Compute gradient
  end for
  $\Delta^{(j)} \leftarrow \frac{1}{L} \sum \Delta_i^{(j)}$   // Average the gradient
  $\Theta \leftarrow \Theta + \lambda\Delta^{(j)}$   // Apply gradient step
end for
return $\Theta$
```

3.4 M-Step. Since the E-step is based on the Markov Chain Monte Carlo sampling, we use the stochastic gradient ascent method to update the parameter matrix Θ to maximize the empirical average of the log-likelihood. In each step, we average the gradient of the log-likelihood (Eq. 3.1), with respect to $\Theta^{(t)}$ and compute the new $\Theta^{(t+1)}$. In this sense, each gradient update eventually repeats the E-step. Moreover, we use the ECM algorithm [25] where only few steps of the gradient ascent are needed for the EM algorithm to converge.

Algorithm 2 gives the M-step where we iterate over the samples of Z and σ obtained in E-step and in each step we compute the gradient of $P(G, Z, \sigma|\Theta)$ as further described in the extended version of the paper [14].

4 Experiments

Next we evaluate the KronEM algorithm on several synthetic and real datasets. First, we use the synthetic Kronecker graphs to examine the correctness of the KronEM algorithm. We test whether KronEM is able to recover the true Kronecker parameters regardless of the fraction of missing nodes. Second, we compare KronEM to link-prediction and the state of the art missing link inference methods on two real datasets, where both nodes and edges are missing as well as where only edges are missing. We show that KronEM compares favorably in both settings. Furthermore, we also examine several aspects of the robustness of KronEM. We show convergence, sensitivity to parameter initialization, and resilience to noise. We also show that KronEM performs well even when the network does not strictly follow the Kronecker graphs model.

4.1 Correctness of KronEM. First we evaluate KronEM on the synthetic Kronecker graphs to check how well KronEM recovers the true Kronecker parameters in presence of missing nodes and edges. Since we know the true parameters that generated the network, we measure the error of Kronecker parameter recovery as a function of the amount of missing data.

We use the Kronecker graphs model with parameter matrix Θ^*, generate a synthetic network H^* on 1,024 nodes, and delete a random set of nodes Z^*. We then use the remaining G^* and infer the true parameter Θ^* as well as the edges in the missing part Z^*. We refer to inferred Z as $\hat{Z}$ and to estimated parameters as $\hat{\Theta}$. In all experiments, we use a 2×2 Kronecker parameter matrix since it has been shown that this already reliably models the structure of real-world networks [17].

Evaluation metrics. We define different distance metrics between two Kronecker graphs by measuring discrepancy between the true parameter matrix Θ^* and the recovered parameter matrix $\hat{\Theta}$. We use L1 (entry-wise sum of absolute differences) and L2 (entry-wise sum of squared differences) distances between Θ^* and $\hat{\Theta}$. We refer to these measures as D_{ABS} and D_{RMS} and compute them as follows: $D_{ABS} = \frac{1}{N_0^2}|\hat{\Theta} - \Theta^*|_1$ and $D_{RMS} = \sqrt{\frac{1}{N_0^2}||\hat{\Theta} - \Theta^*||_2}$. We also consider Kullback-Leibler (KL) divergence that operates directly on the full stochastic adjacency matrices constructed by Θ^* and $\hat{\Theta}$. Naive computation of KL divergence takes $O(N^2)$ time, however, we can approximate it by exploiting the recursive structure of Kronecker graphs. By using Taylor expansion $\log(1-x) \approx -x - 0.5\ x^2$, we obtain $D_{KL} \approx k|\Theta^*|^{k-1}\left(\sum_{i,j}\Theta^*_{ij}\log\left(\Theta^*_{ij}/\hat{\Theta}_{ij}\right)\right) - |\Theta^*|^k + \frac{1}{2}S(\Theta^*)^k + |\hat{\Theta}|^k + \frac{1}{2}S(\Theta^*)^k - \left(\sum_{i,j}\Theta^*_{ij}\hat{\Theta}_{ij}\right)^k$ where $|\Theta| = \sum_{i,j}\Theta_{ij}$ and $S(\Theta) = \sum_{i,j}\Theta^2_{ij}$

Finally, we also directly measure the log-likelihood as a generic score of the similarity between an instance and a distribution. Since we can additionally calculate the log-likelihood of the true Θ^*, we consider the difference of log-likelihood between the true and the estimated Kronecker parameters as a relative likelihood measure. We further distinguish two ways of quantifying these differences. One is the difference of the likelihood between the true complete network H^* and the estimated one $\hat{H}$. The other is that between the missing parts of networks Z^* and $\hat{Z}$. We therefore define $D_{LL} = LL(\hat{\Theta}) - LL(\Theta^*)$ and $D_{LL_Z} = LL_Z(\hat{\Theta}) - LL_Z(\Theta^*)$.

Method	D_{ABS}	D_{RMS}	D_{KL}	D_{LL}	D_{LL_Z}
KronFit	0.173	0.048	354.4	-355	-155
KronEM	0.063	0.018	22.4	-12	-5

Table 1: Performance on a synthetic Kronecker graph. Notice KronEM outperforms KronFit in all respects.

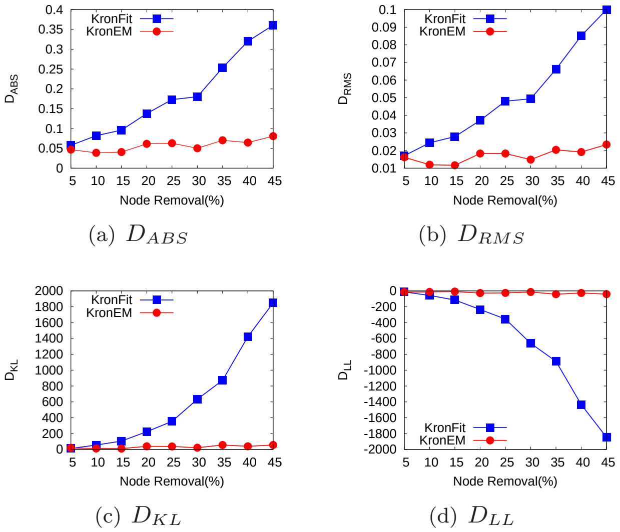
Results on synthetic data. We compare KronEM to fitting Kronecker parameters directly to the observed part G^* by inserting isolated nodes in the unobserved part Z (we refer to the method as KronFit [19]). Note that this method ignores the missing part and estimates the model parameters purely from the observed G^*.

Table 1 shows the performance of the KronEM and KronFit where we randomly removed 25% of the nodes so that the network has 768 nodes and 1,649 edges (out of 1,024 nodes and 2,779 edges). Notice that KronEM greatly outperforms KronFit in all respects. Both the L1 (D_{ABS}) and L2 (D_{RMS}) distances of the estimated parameters are more than twice smaller for KronEM. In terms of the KL divergence and the difference in the log-likelihood we see about factor 20 improvement. These experiments demonstrate that the Expectation Maximization algorithm greatly outperforms Kronecker parameter fitting directly to the observed data with ignoring the missing part.

We additionally examine the inference performance as a function of the amount of missing data. KronEM maintains performance as the number of unobserved data (*i.e.*, the size of Z) increases. For this purpose, we vary the fraction of missing nodes and measure the error metrics, D_{ABS}, D_{RMS}, D_{KL} and D_{LL}. Figure 2 gives the results of these experiments. Notice that the error of KronEM is nearly flat and is increasing even slower than the error of KronFit. When we remove up to 45% of the nodes, KronEM is still able to reliably estimate the model parameters and recover the missing part of the network. On the other hand, KronFit becomes infeasible to recover the parameter already when only about 10% of the data is missing.

To sum up, KronEM near-perfectly recovers the true Kronecker parameters with less than 0.02 D_{RMS} error, regardless of the amount of missing data.

4.2 Experiments on Real Networks. From the experiments so far, we see that when a network follows the Kronecker graphs model KronEM can accurately



(a) D_{ABS}

(b) D_{RMS}

(c) D_{KL}

(d) D_{LL}

Figure 2: Error of the recovered missing part of the network as a function of the fraction of missing nodes in the network. KronEM is able to reliably estimate the missing part even with 45% of the nodes missing.

solve the network completion problem, even though a large part of the network is missing. Now we turn our attention to real datasets and evaluate our approach KronEM on these real networks. In the evaluation of KronEM, focusing on the inference of missing edges as well as the recovery of global network properties, we compare KronEM to other methods.

For these experiments, we use two real-world networks: a part of the Autonomous System (AS) network of the Internet connectivity (4,096 nodes, 29,979 edges) [20] and a part of the Flickr social network (4,096 nodes, 22,178 edges) [16]. Then, we randomly removed 25% of the nodes and the corresponding edges.

Baseline methods. To assess the quality of KronEM for network completion problem, we introduce different classes of methods and compare them with KronEM.

First, we consider a set of classical link prediction methods [21]. We choose two link prediction methods that have been found to perform well in practice [21]: Degree-Product (DP), and Adamic-Adar score (AA). We select the Degree-Product because it models the preferential attachment mechanism [2], and the Adamic-Adar score since it performs the best in practice [21]. Adamic-Adar score can be regarded as an extension of the Jaccard score that considers the number of common neighbors but gives each neighbor different weight.

We also consider another model-based approach to network completion. We adopt the Stochastic block model (BM) that is currently regarded as the state of the art as the method already demonstrated good performance for missing link inference [10]. The block model assumes that each node belongs to a latent group and the relationship between the groups governs the connections. Specifically, a group membership $B(u)$ for every node u and the group relationship probability $P_{BM}\left(B(u), B(v)\right)$ for every pair of groups are defined in the model, and a pair of nodes u and v is connected with this group relationship probability $P_{BM}\left(B(u), B(v)\right)$. For the inference task with the Stochastic block model, we implemented the algorithm described in [10].

For every edge (every pair of nodes), we compute the score of the edge under a particular link prediction method and then consider the probability of the edge being proportional to the score. For each method, the score of an edge (x, y) is:

- Degree Product (DP) : $Deg(x) \cdot Deg(y)$
- Adamic-Adar (AA) : $\sum_{z \in Nb(x) \cap Nb(y)} \frac{1}{\log Deg(z)}$
- Block Model (BM) : $P_{BM}\left(B(x), B(y)\right)$

where $Deg(x)$, $Nb(x)$, and $B(x)$ denote the degree of x, a set of neighbors of x, and the corresponding group of x in the block model, respectively. In this way, each method generates a probability distribution over the edges in the unobserved part of the network Z.

Evaluation. We evaluate each algorithm on its ability to infer the missing part of the network. Since each method (DP, AA, BM, and KronEM) is able to compute the probability of each edge candidate, we quantify the performance of inference by the area under ROC curve (AUC) and the log-likelihood (LL). Moreover, we are particularly interested in evaluating the performance over the part where both endpoints of an edge are in the missing part of the network. We denote those values as AUC_{N_Z} and LL_{N_Z} respectively.

Table 2 shows the results for this inference task. Our proposed approach of KronEM gives a significant boost in performance compared to DP and AA – it increases the AUC score for about 20%. Comparison with BM shows better performance in AUC. Moreover, KronEM performs particularly well when estimating the edges between the missing nodes (AUC_{N_Z}). In this AUC_{N_Z} measure, KronEM outperforms BM by $10 \sim 15\%$. Furthermore, we verify the quality of each ROC curve associated with AUC and AUC_{N_Z} in Figure 3 which shows that KronEM produces better ROC curves than the other methods for both datasets.

In the other scores, LL and LL_{N_Z}, KronEM also performs better than DP and AA. On the other hand, BM produces better LL scores than KronEM. However, when we take account of the number of parameters in the model (2,916 for BM and only 4 for KronEM), AS network

Method	AUC	AUC_{N_Z}	LL	LL_{N_Z}
DP	0.661	0.5	-86,735	-14,010
AA	0.634	0.701	-106,710	-31,831
BM	0.834	0.737	-73,103	-11,881
KRONEM	0.838	0.800	-76,571	-12,265

Flickr network

Method	AUC	AUC_{N_Z}	LL	LL_{N_Z}
DP	0.782	0.5	-75,921	-11,324
AA	0.706	0.900	-81,089	-18,676
BM	0.875	0.806	-50,124	-7,972
KRONEM	0.930	0.925	-54,889	-7,841

Table 2: The performance of inference by each method on AS and Flickr networks: KRONEM performs best in AUC and AUC_{N_Z}. BM seems better in LL and LL_{N_Z}, however a large number of parameters in BM lead to overfitting.

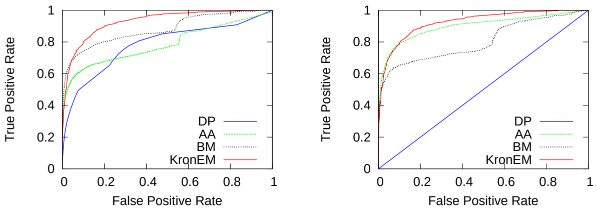


Figure 3: ROC curves for Flickr network: KRONEM produces better ROC curves in the entire missing part Z (left) as well as in the missing part between only missing nodes (right).

this phenomenon seems natural because larger degree of freedom is likely to overestimate the likelihood. In case that we limit the number of parameters in BM to 900 not to overestimate the likelihood, we observe that AUC and LL for the Flickr network decrease to 0.772 and -66,653 respectively.

In these experiments, we see that the model-based approaches (KRONEM and Stochastic block model) outperform the classical link-prediction methods (Adamic-Adar and Degree-Product). Between the two model-based approaches, KRONEM is superior to BM. Focusing on the part between missing nodes and considering the number of parameters, KRONEM shows better performance than the Stochastic block model.

Recovery of global network properties. In the previous experiments, we examined the performance of the inference task on individual edges and showed that KRONEM is the most successful in terms of the AUC. Along with these microscopic edge inference tests, we are also interested in the recovery of global network properties such as degree distribution or local edge clustering. In other words, we view the results of inference in a macroscopic way rather than a microscopic manner.

Instead of comparing the complete networks $\hat{H}$ and H^*, *i.e.*, the observed and the unobserved parts together, we focus only on the hidden parts Z^* and $\hat{Z}$. That is, we look at the overall shape of the networks only over the missing part. We then examine the following properties:

- *In/Out-Degree distribution (InD/OutD)* is a histogram of the number of in-coming and out-going links. Networks have been found to have power-law degree distributions [27].
- *Singular values (SVal)* indicate the singular values of the adjacent matrix versus their ranks. This plot also tends to be heavy-tailed [7].
- *Singular vector (SVec)* represents the distribution of components in the eigen vector associated with the largest eigen value. It has been also known to be heavy tailed [5].
- *Triad Participation (TP)* indicates the number of triangles that a node participates in. It measures the transitivity in networks [29].

To quantify the similarity between the recovered and the true statistical network property, we apply a variant of Kolmogorov-Smirnov (KS) statistic to quantify a distance between two distributions in a non-parametric way. Since the original KS statistic may be dominated by few extreme values (which are common to occur when working with heavy-tailed distributions), we adopt a modified KS static $PowerKS(\hat{F_1}, \hat{F_2})$ for two empirical cumulative distributions, where $PowerKS(\hat{F_1}, \hat{F_2}) = \sup_x |\log(1 - F_1(x)) - \log(1 - F_2(x))|$. With this statistic, we capture the difference of distributions in the log-log scale. Moreover, by using the complementary cumulative distribution, we maintain the linear shape of the power-law distribution in the log-log plot, but still remove the noisy effect in the distribution as the cumulative distribution does.

Table 3 shows the *PowerKS* statistic values of various network properties for KRONEM as well as the other methods. On the average, KRONEM gives 40% reduction in the *PowerKS* statistic over the Degree-Product (DP) and 70% reduction over the Adamic-Adar (AA) method on both datasets. Furthermore, KRONEM represents better *PowerKS* statistics than the Stochastic block model (BM) in 3 out of 5 faces on both datasets. From the results, we can see that KRONEM also performs well in the recovery of global properties.

Recovery as a function of the amount of missing data. Next we check the performance change of each algorithm as a function of the amount of missing data.

AS network

Method	InD	OutD	SVal	SVec	TP
DP	1.99	2.00	0.19	0.04	4.24
AA	1.89	1.89	3.45	0.05	4.08
BM	2.05	2.04	0.42	0.21	2.20
KronEM	2.14	2.20	0.17	0.08	1.81

Flickr network

Method	InD	OutD	SVal	SVec	TP
DP	1.77	1.80	0.75	0.21	5.77
AA	2.40	2.23	8.46	0.18	1.00
BM	1.89	2.03	1.02	0.15	1.74
KronEM	1.71	1.62	0.38	0.21	3.49

Table 3: PowerKS statistics for AS network and Flickr network: Overall, KronEM recovers the global network properties most accurately.

This indirectly tells us how dependent on the observed information each algorithm is.

For these experiments, we used the Flickr network and removed 5%, 25%, 45% of nodes randomly. We note that each of the node-removal fraction results in approximately 10%, 43%, 70% edges removed, *i.e.*, by removing 45% of the nodes, we lose 70% of the edges.

In spite of this high fraction of missing edges, KronEM shows sustainable performance. First of all, in Figure 4, there exists almost no drop in *AUC* results of KronEM, while *AUC*'s of the other methods are continuously decreasing as the amount of missing data increases. The performance of the Adamic-Adar (AA) is the most hurt by the missing data. Interestingly, the Block Model (BM) suffers more than the Degree Product (DP) (Figure 4(right)). On the other hand, the performance of KronEM practically does not decrease even with the amount of missing data as high as 40%.

Furthermore, when taking a careful look at how KronEM recovers global network properties, there exists little difference between the true and the inferred networks as shown in Figure 5. When the number of missing nodes is lower, we get a better fit, but the gap does not increase a lot as the missing fraction increases.

4.3 Inference of Missing Edges. We so far performed the experiments in the environment where some fraction of nodes and their corresponding edges are missing. However, we also consider a slightly different problem that all nodes are observed but a fraction of random edges are missing. We can view this modified setting as a class of missing link inference problem. Since the baseline methods (Degree-Product, Adamic-Adar, and Stochastic block model) have usually been applied to this type of setting, we also want to inspect the performance of KronEM in this situation.

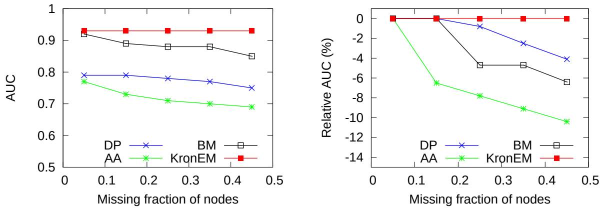
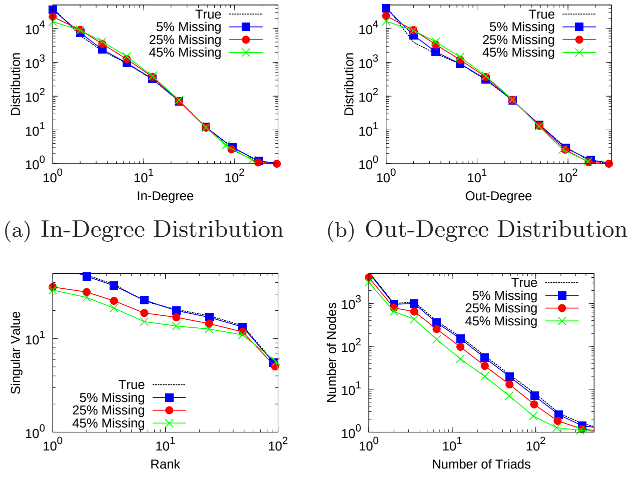


Figure 4: *AUC* change of inference on the Flickr network for various missing fractions of nodes: Both absolute value (left) and relative value (right) illustrate that KronEM maintains its high performance, while the accuracies of the other methods drop.



(a) In-Degree Distribution

(b) Out-Degree Distribution

(c) Singular Value

(d) Triad Participation

Figure 5: Recovery of the global properties on the Flickr network for various fractions of missing nodes: Inferred networks according to the fraction of missing part do not show a large difference each other, even though the fraction of missing edges is up to 70%.

Using the same real datasets (AS and Flickr), we selected 25% of edges at random and removed them. We then examine the inference accuracy of each algorithm under the *AUC* metric as before.

Table 4 shows the *AUC* results of all methods for both real networks. Note that the *AUC* scores of methods except for KronEM increase compared to the scores in the inference of missing nodes and edges. Because Degree-Product (DP) and Adamic-Adar (AA) depend on the local information of each node, their performances show large improvement between the two different inference tasks. On the other hand, since Stochastic block model (BM) and KronEM are model-based approaches, these methods reduce the performance loss caused by the lack of the node information in the previous task where we infer both missing nodes and edges. Even though the classical link-prediction methods rel-

Network	DP	AA	BM	KronEM
AS	0.750	0.760	0.821	0.833
Flickr	0.815	0.848	0.895	0.900

Table 4: *AUC* results for missing edge inference on AS and Flickr networks. Even though the other methods become more competitive in this setting than in the inference of both missing nodes and edges, KronEM still performs best.

atively increase their inference performance, KronEM produces the best inferences in both networks, and even slightly outperforms the Stochastic Block Model.

4.4 Robustness of KronEM. So far we have focused on the two aspects of the performance of KronEM: the recovery power of Kronecker parameters and the ability in both individual edge inference and recovery of the global properties with real datasets.

Now we move our focus to the robustness of KronEM. We define the robustness mainly in two ways. First, we consider the robustness of the algorithm itself in terms of convergence, sensitivity to the initializations, and resilience to the noise. Second, we also evaluate the robustness against the underlying network structure in a sense of how good performance KronEM yields even when the network does not strictly follow the Kronecker graphs model. Although the experiments on real datasets already demonstrate this in some sense, we verify this robustness by using synthetic networks generated by various kinds of social network models and show that the Kronecker graphs structure is not the necessary condition for KronEM to work well.

Robustness of algorithm. We categorize the factors that affect on the final solution into three groups.

First, to check the variance of the randomized algorithm, we run KronEM several times by setting different random starting points over the synthetic network. Figure 6(a) shows the convergence of parameter Θ_{01} as a function of the number of EM iterations for each run. Notice that when the current parameter estimate is far from the optimal one, the algorithm quickly converges to good parameter values. On the other hand, as the parameter estimate moves close to the optimal value, the random effect in the algorithm emerges. However, all the points move toward the same final solution with keeping their variance small.

Second, to see the effect of the parameter initialization conditions, we took several experiments with different starting points. We run KronEM from a random starting parameter matrix Θ over the networks with parameters $\Theta^* = [0.9\ 0.6;\ 0.4\ 0.2]$ and 25% of the nodes missing. Since KronEM solves a non-convex problem, the algorithm may converge to local optimal solutions for the missing part Z, the node mapping σ, and the Kronecker parameter Θ. In practice, we notice that this might not be a problem in the following sense. Figure 6(b) illustrates the convergence of the parameter Θ_{01} over the number of iteration from initial random starting points. The curves begin with a wide range of initial values, but eventually converge into only two values. Moreover, in this Figure 6(b), while large initial points (larger than 0.3) converge to the larger final value (0.6), small initial points (smaller than 0.3) converge to the smaller final value (0.4). Notice that these are exactly the two off-diagonal values of the parameter matrix that we used to generate the graph. Since Kronecker parameter matrix is invariant to permutations of entries, this means that KronEM is able to exactly recover the parameters regardless of the initialization conditions. Overall, we observe that KronEM algorithm is not sensitive to the random initialization settings. Therefore, when finding the optimal Θ, we run KronEM several times with different parameter initializations and select the solution with maximum likelihood.

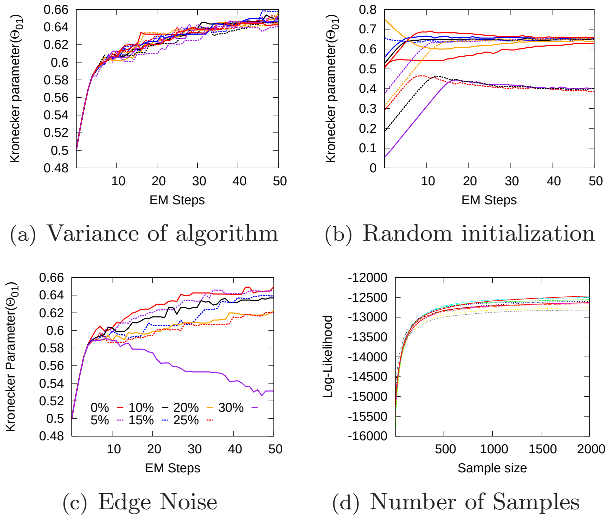
Third, in order to check how robust KronEM is against the edge noise, we added various rates of random noise into the edges in the observed part of the graph. We randomly picked a number of edges and the same number of non-edges, and flipped their values (*i.e.*, edges become non-edges, and non-edges become edges). This means that we are imposing the Erdős-Rényi random graph over the existing network. As real-networks do not follow Erdős-Rényi random graph model, this process heavily distorts the network.

Figure 6(c) shows the estimate of Kronecker parameter Θ_{01} for different edge noise rates. Notice that KronEM results in almost the same solution against up to 15% edge noise rate. Even at 20% and 25% noise, the estimates are acceptable since their differences from the original are less than 0.02. As the amount of noise increases beyond 25%, we start to observe a significant error in the parameter estimation.

Additionally, we investigate the number of samples of Z and σ needed before the empirical estimates of the log-likelihood and the gradient converge. Figure 6(d) shows the convergence of average likelihood in one E-step for the network 1,024 nodes. Notice that only about 1,000 samples are needed for the estimate to converge.

Based on these experiments, we conclude that KronEM exhibits low variance, low sensitivity to the initialization conditions, and high resilience against the edge noise.

Robustness against the underlying network model. Last we also investigate the performance of KronEM even when the underlying network does not follow the Kronecker graph model. We generated syn-



(a) Variance of algorithm (b) Random initialization

(c) Edge Noise (d) Number of Samples

Figure 6: Convergence and robustness of algorithm: KronEM has low variance, low sensitivity to the parameter initialization, and high resilience to the noise.

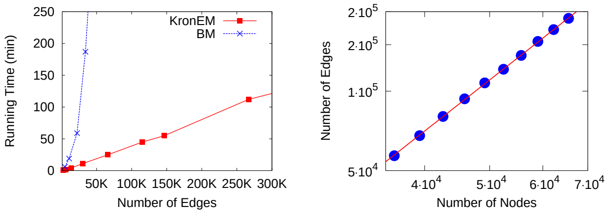
thetic networks using several models: Preferential Attachment [27], Forest Fire [20], Kronecker Graphs [18] and Stochastic Block Model. We then performed the same experiments for real datasets. We randomly removed 25% of nodes in the synthetic networks, computed the probability of each edge candidate using different methods (AA, DP, BM and KronEM), and measured the results in terms of the area under ROC curve (AUC, AUC_{N_Z}) and the log-likelihood (LL, LL_{N_Z}). These experiments are particularly interesting because the synthetic networks generated by different models do not follow the Kronecker graphs model but resemble the real-world networks in some properties.

Table 5 shows the AUC score for different synthetic networks. (We omit the other scores for brevity.) Notice that on all networks KronEM gives the best results. KronEM as well as the other methods perform the best on Forest Fire and Kronecker graphs, while the network completion in Preferential Attachment and Stochastic Block Model graphs seems to be a harder problem. Overall, KronEM outperforms the classical link-prediction methods (DP and AA), and shows better performance than the Stochastic block model (BM) regardless of underlying network structure. In summary, even though these networks do not follow the Kronecker graphs model, KronEM still makes the most accurate inferences.

4.5 Scalability of KronEM. Next we briefly present the empirical analysis of KronEM running time and examine how it scales as we increase the size of the network H. We generated a sequence of Kronecker graphs with $N_0 = 2$ and $k = 10, 11, \ldots, 16$, i.e, each network has 2^k nodes. As before, we removed 25% of random nodes for each network. Figure 7(a) shows the dependency between the network size and running time. We compare the KronEM running time with the Stochastic Block Model approach (BM) [10] for network completion. KronEM scales much better than BM. Empirically KronEM runtime scales as $O(E \log E)$ with number of edges E in the network.

Network	DP	AA	BM	KronEM
Kron	0.749	0.833	0.862	0.916
PA	0.600	0.660	0.751	0.779
FF	0.865	0.769	0.743	0.946
SBM	0.663	0.641	0.772	0.796

Table 5: Inference performance (AUC) on synthetic networks generated using different network generative models (Kronecker graph (Kron), Preferential Attachment (PA), Forest Fire (FF) and Stochastic Block Model (SBM)). Performance of KronEM is stable regardless of the model used to generate the underlying synthetic network.



(a) Running time (b) Edges vs. nodes

Figure 7: (a) Running time as a function of network size: the running time of KronEM fits to $y = ax \log x$, while that of BM increases quadratically. (b) The number of observed edges as a function of the missing nodes follows quadratic relation represented by the solid line, $y = ax^2$.

4.6 Estimating the Number of Missing Edges. So far we operated under the assumption that we know the number of missing nodes as well as missing edges. However, in practice we may only know the number of missing node, but not the number of missing edges.

In order to estimate the number of missing edges E_Z, we use the following approach. In the case of random node removal, we observe that the number of observed edges is roughly proportional to the square of the number of observed nodes. This holds because the degrees of the removed nodes follow the same distribution as those of the full network. In addition, the relationship between the number of observed nodes and edges with varying the missing fraction is further verified in Figure 7(b). Thus, we can compute the number of missing edges $\mathbb{E}_{\mathbb{Z}}$ as $\mathbb{E}(E_Z) \approx \frac{1-(1-e)^2}{(1-e)^2} E_G$ where e is the fraction of missing nodes, $e = N_Z/N$.

5 Conclusion

We investigated the network completion problem where the network data is incomplete with nodes and edges missing or being unobserved. The task is then to infer the unobserved part of the network. We developed KronEM, an EM approach combined with the Kronecker graphs model, to estimate the missing part of the network. As the original inference problem is intractable, we use sampling techniques to make the estimation of the missing part of the network tractable and scalable to large networks. The extensive experiments suggest that KronEM performs well on synthetic as well as on real-world datasets.

Directions for future work include extending the present approach to the cases where rich additional information about node and edge attributes is available.

Acknowledgments. Research was supported in part by NSF grants CNS-1010921, IIS-1016909, LLNL grant DE-AC52-07NA27344, Albert Yu & Mary Bechmann Foundation, IBM, Lightspeed, Microsoft and Yahoo.

References

[1] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. *JMLR*, 9:1981–2014, 2008.

[2] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[3] K. Bleakley, G. Biau, and J. Vert. Supervised reconstruction of biological networks with local models. *Bioinformatics*, 23(13):i57–i65, 2007.

[4] E. J. Candes and B. Recht. Exact matrix completion via convex optimization. *Found. Comput. Math.*, 9(6):717–772, 2009.

[5] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-mat: A recursive model for graph mining. In *SDM*, 2004.

[6] A. Clauset, C. Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, May 2008.

[7] I. J. Farkas, I. Derényi, A.-L. Barabási, and T. Vicsek. Spectra of "real-world" graphs: Beyond the semicircle law. *Phys. Rev. E*, 64(2):026704, Jul 2001.

[8] D. Goldberg and F. Roth. Assessing experimentally derived interactions in a small world. *Proceedings of the National Academy of Sciences*, 100(8):4372, 2003.

[9] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In *KDD '10*, 2010.

[10] R. Guimerá and M. Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. *PNAS*, 106(52), 2009.

[11] S. Hanneke and E. Xing. Network completion and survey sampling. In *AISTATS '09*, 2009.

[12] R. H. Keshavan, S. Oh, and A. Montanari. Matrix completion from a few entries. In *ISIT '09*, pages 324–328, 2009.

[13] P. Killworth, C. McCarty, H. Bernard, G. Shelley, and E. Johnsen. Estimation of seroprevalence, rape, and homelessness in the United States using a social network approach. *Evaluation Review*, 22(2):289, 1998.

[14] M. Kim and J. Leskovec. Network completion problem: Inferring missing nodes and edges in networks. *http://www.stanford.edu/∼mykim/paper-kronEM.pdf.*

[15] G. Kossinets. Effects of missing data in social networks. *Social Networks*, 28:247–268, 2006.

[16] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins. Microscopic evolution of social networks. In *KDD '08*, pages 462–470, 2008.

[17] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. *JMLR*, 2010.

[18] J. Leskovec, D. Chakrabarti, J. M. Kleinberg, and C. Faloutsos. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In *PKDD '05*, pages 133–145, 2005.

[19] J. Leskovec and C. Faloutsos. Scalable modeling of real graphs using kronecker multiplication. In *ICML '07*, 2007.

[20] J. Leskovec, J. M. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD '05*, 2005.

[21] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM '03*, pages 556–559, 2003.

[22] M. Mahdian and Y. Xu. Stochastic kronecker graphs. In *WAW '07*, 2007.

[23] P. Marsden. Network data and measurement. *Annual review of sociology*, 16(1):435–463, 1990.

[24] T. H. McCormick and M. J. Salganik. How many people you know?: Efficiently esimating personal network size, September 17 2008.

[25] X.-L. Meng and D. B. Rubin. Maximum likelihood estimation via the ecm algorithm: A general framework. *Biometrika*, 80(2):267–278, 1993.

[26] S. Myers and J. Leskovec. On the convexity of latent social network inference. In *NIPS '10*, 2010.

[27] Price and D. J. de S. A general theory of bibliometric and other cumulative advantage processes. *Journal of the American Society for Information Science*, 27(5-6):292–306, 1976.

[28] B. Taskar, M. F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *NIPS '03*, 2003.

[29] C. E. Tsourakakis. Fast counting of triangles in large real networks without counting: Algorithms and laws. *ICDM*, 2008.

[30] G. C. G. Wei and M. A. Tanner. A monte carlo implementation of the em algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85:699–704, 1990.

[31] Y. Yamanishi, J. Vert, and M. Kanehisa. Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, 20:i363–i370, 2004.

A APPENDIX: Mathematics of KronEM

In this section, we describe the mathematical details of KronEM that we described in Section 3. First, we derive the log-likelihood formula $\log P(G, Z, \sigma|\Theta)$ and its gradient $\frac{\partial P(G,Z,\sigma|\Theta)}{\partial \Theta_{ij}}$, where G, Z, σ, and Θ represent the observed network, the missing part of the network, the node mapping, and Kronecker parameter matrix, respectively (*i.e.*, Θ_{ij} is the $(i,\ j)$ entry of Θ). Second, we present the proposal mechanism and its detailed balance equation of Metropolis-Hastings sampling from $P(Z|G, \sigma, \Theta)$. Finally, we provide the proposal mechanism and the detailed balance equation of Metropolis sampling from $P(\sigma|G, Z, \Theta)$.

A.1 Log-likelihood Formula. We begin with showing $P(H|\sigma, \Theta)$ for a complete Kronecker graph $H = G \cup Z$.

$$P(H|\sigma,\Theta) = \prod_{u\sim v} [\Theta^k]_{\sigma(u)\sigma(v)} \prod_{u\not\sim v} (1 - [\Theta^k]_{\sigma(u)\sigma(v)})$$

Since $P(G, Z, \sigma|\Theta) = P(H|\sigma, \Theta)P(\sigma|\Theta)$, if we regard σ as being independent of Θ when the network H is unknown and the prior distribution of σ is uniform, then we obtain

$$P(G, Z, \sigma|\Theta) \propto P(H, \sigma, \Theta)$$

Therefore, when computing the likelihood, we use the Equation (3.1). If we take the log-value of Equation (3.1) and remove a constant factor, we define the equivalent log-likelihood value $LL(\Theta)$ as follows:

(A.1)
$$\begin{aligned} LL(\Theta) &= \log P(G, Z, \sigma|\Theta) + C \\ &= \sum_{u\sim v} \log [\Theta^k]_{\sigma(u)\sigma(v)} + \sum_{u\not\sim v} \log(1 - [\Theta^k]_{\sigma(u)\sigma(v)}) \\ &= \sum_{u,v} \log(1 - [\Theta^k]_{\sigma(u)\sigma(v)}) - 2\sum_{u\sim v} \log [\Theta^k]_{\sigma(u)\sigma(v)} \end{aligned}$$

However, it takes quadratic time in the number of nodes N to calculate this $LL(\Theta)$, because we should compute the log-likelihood for every pair of nodes. Since this computation time is not acceptable for large N, we develop a method of approximating the $LL(\Theta)$. By Taylor's expansion, $\log(1 - x) \approx -x - 0.5x^2$ for small x. Plugging this formula into $\log(1 - [\Theta^k]_{\sigma(u)\sigma(v)})$,

$$\begin{aligned} &\sum_{u,v} \log(1 - [\Theta^k]_{\sigma(u)\sigma(v)}) \approx \\ &\qquad - \sum_{u,v} \left([\Theta^k]_{\sigma(u)\sigma(v)} + 0.5([\Theta^k]_{\sigma(u)\sigma(v)})^2\right) \end{aligned}$$

Since $\sigma(u)$ permutates $0 \sim {N_0}^k$,

$$\text{(A.2)}\quad \sum_{u,v} \log(1 - [\Theta^k]_{\sigma(u)\sigma(v)}) \approx -|\Theta|_1^k - 0.5\left(||\Theta||_2^2\right)^k$$

where $|\Theta|_1$ and $||\Theta||_2^2$ denote the sum of absolute and squared values of entries, respectively. Thus, by combining Equation (A.1) and (A.2), we are able to approximately compute $LL(\Theta)$ in $O(E)$ time for the number of edges E.

Furthermore, by this combination of equations, we also derive the approximate gradient of $LL(\Theta)$:

$$\text{(A.3)}\qquad \begin{aligned} \frac{\partial LL}{\partial \Theta_{ij}} &\approx -k|\Theta|_1^{k-1} - k\Theta_{ij}||\Theta||_2^{k-1} \\ &\qquad - 2\sum_{u\sim v} \frac{1}{\Theta_{ij}} C_{ij}(\sigma(u), \sigma(v)) \end{aligned}$$

where $C_{ij}(\sigma(u), \sigma(v))$ denotes the number of bits such that $\sigma(u)_l = i$ and $\sigma_(v)_l = j$ for $l = 1, 2, \cdots, k$ when $\sigma(u)$ and $\sigma(v)$ are expressed in N_0-base representation. Therefore, computing each gradient component also takes $O(E)$ time.

A.2 Metropolis-Hastings Sampling of Z. In the main paper, we briefly described the algorithm for sampling the missing part Z where the node mapping σ and the Kronecker parameter Θ are provided. Here we present the proposal mechanism and the corresponding detailed balance condition to determine the acceptance rate.

We consider the following simple mechanism. Given the previous sample of the missing part Z, we select an edge in Z at random and remove it. Then, we select a pair of nodes in Z which is not connected in the current state proportionally to its edge probability, and add it as an edge of the new sample Z'. When adding this new edge, we use the same acceptance-rejection algorithm described in Section 3.

In order to check the detailed balance condition for $Z \neq Z'$, ignoring irrelevant factors (*e.g.*, σ) for simplicity, we derive the transition probability:

$$\begin{aligned} &P(Z)P(Z \to Z') \\ &= P(Z\backslash x, y)P(x \in E_Z)P(y \notin E_Z)P(\text{del } x, \text{add } y) \\ &= P(Z\backslash x, y)P(x)(1 - P(y))\frac{1}{|E_Z|}\frac{P(y)}{\sum_{e\notin E_Z} P(e) + P(x)} \end{aligned}$$

where E_Z denotes the set of edges in the missing part Z and $P(x)$ represents the probability of an edge x for fixed Θ and σ. Similarly,

$$\begin{aligned} &P(Z')P(Z' \to Z) \\ &= P(Z'\backslash x, y)P(y)(1 - P(x))\frac{1}{|E_{Z'}|}\frac{P(x)}{\sum_{e\notin E_{Z'}} P(e) + P(y)} \end{aligned}$$

Algorithm 3 SampleZ : Sample the new sample of Z

```
input: observed graph $G$, Kronecker parameter $\Theta$
       node mapping $\sigma$, current sample $Z$
output: new sample $Z'$

$E_Z \leftarrow \{z \in Z | z : \text{edge}\}$
$NE_Z \leftarrow \{z \in Z | z : \text{non edge}\}$
$x \leftarrow$ an uniformly random sample from $E_Z$
$E_Z \leftarrow E_Z \backslash \{x\}$
$NE_Z \leftarrow NE_Z \cup \{x\}$
$y \leftarrow$ a random sample from $NE_Z$ proportionally to $P(y|\Theta, \sigma)$
$u \sim U(0, 1)$
if $u < \min\left(1, \frac{1-P(y|\Theta,\sigma)}{1-P(x|\Theta,\sigma)}\right)$ then
    $E_Z \leftarrow E_Z \cup \{y\}, NE_Z \leftarrow NE_Z \backslash \{y\}$
else
    $E_Z \leftarrow E_Z \cup \{x\}, NE_Z \leftarrow NE_Z \backslash \{x\}$
end if
$Z' \leftarrow$ Combine $E_Z$ and $NE_Z$
return $Z'$
```

Since $Z \backslash x, y = Z' \backslash x, y$ and $|E_Z| = |E_{Z'}|$, the ratio of the transition probabilities for the two directions is derived by nicely canceling out many terms:

$$\frac{P(Z')P(Z' \to Z)}{P(Z)P(Z \to Z')} = \frac{1 - P(x)}{1 - P(y)} \tag{A.4}$$

Due to the asymmetric transition probability, Equation (A.4) has to be introduced as the correction factor in Metropolis-Hastings algorithm. In the result, we accept the new sample Z' with the following acceptance probability; otherwise, we reject the new sample and maintain the current sample Z.

$$\mathcal{A}(Z' \to Z) = \min\left(1, \frac{1 - P(y)}{1 - P(x)}\right)$$

Algorithm 3 shows the overall process of this Metropolis-Hastings sampling for the missing part Z. For an additional advantage of this process, we can update the log-likelihood or the gradient of the log-likelihood in very short time ($O(k)$).

A.3 Metropolis Sampling of σ. Here we present the sampling algorithm for the node mapping σ. As decribed in the main paper, we develop a Metropolis sampling that updates the new sample σ' given the current sample σ. For the update $\sigma \to \sigma'$, it turns out that a random node-mapping swapping practically works well [19].

To examine the detailed balance condition of node-mapping swapping (*i.e.*, $\sigma'(u) = \sigma(v), \sigma'(v) = \sigma(u)$ for nodes u, v),

Algorithm 4 SampleSigma : Sample the new sample of σ

```
input: full network $H$, Kronecker parameter $\Theta$
       current node-mapping sample $\sigma$
output: new node-mapping sample $\sigma'$

$u, v \leftarrow$ randomly sample two nodes
$\sigma' \leftarrow \sigma$
$\sigma'(u) \leftarrow \sigma(v)$
$\sigma'(v) \leftarrow \sigma(u)$
$u \sim U(0, 1)$
if $u \geq \min\left(1, \frac{P(H|\sigma',\Theta)}{P(H|\sigma,\Theta)}\right)$ then
    $\sigma' \leftarrow \sigma$        // Roll-back
end if
return $\sigma'$
```

$$\begin{aligned}
&\frac{P(\sigma')P(\sigma' \to \sigma)}{P(\sigma)P(\sigma \to \sigma')} = \frac{P(H|\sigma', \Theta)}{P(H|\sigma, \Theta)} \\
&= \prod_{x \sim u} \frac{P_{\sigma'}(x, u)}{P_\sigma(x, u)} \prod_{x \not\sim u} \frac{1 - P\sigma'(x, u)}{P_\sigma(x, u)} \\
&\qquad \times \prod_{x \sim v} \frac{P_{\sigma'}(x, v)}{P_\sigma(x, v)} \prod_{x \not\sim v} \frac{1 - P\sigma'(x, v)}{P_\sigma(x, v)} \\
&= \left(\prod_{x \sim u} \frac{P_\sigma(x, u)}{P_{\sigma'}(x, u)} \prod_{x \sim v} \frac{P_\sigma(x, v)}{P_{\sigma'}(x, v)} \right)^2
\end{aligned}$$

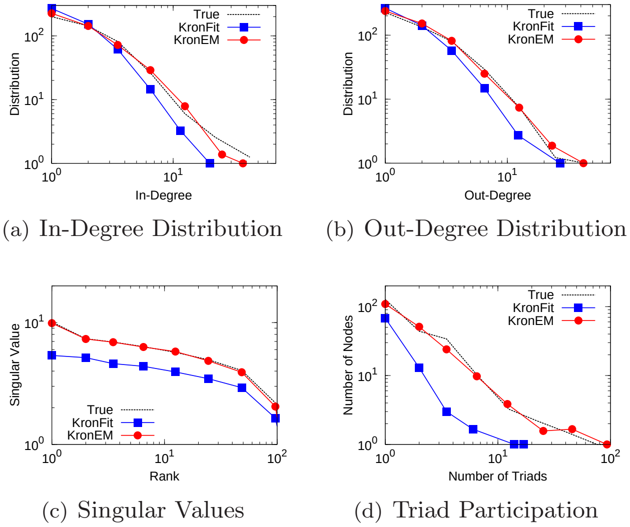
where H and Θ respectively represent the complete network and the Kronecker parameter, and $P_\sigma(x, u) = P(x \sim u | \Theta, \sigma)$.

Applying this ratio to the correction factor for Metropolis sampling, we develop the Algorithm A.3.

B Supplementary Experiments

We already presented some experimental results in Section 4, but we omitted several results due to the lack of space in the main paper. Here we provide additional results of our experiments.

B.1 Correctness of KronEM in the Recovery of Global Network Properties. We showed that KronEM outperforms KronFit in the accuracy of Kronecker parameter recovery. We additionally performed the experiment that compares the performance in the recovery of network global properties on the same synthetic Kronecker graph as in Section 4.1. Figure 8 shows that KronEM almost perfectly recovers the network properties whereas KronFit cannot.



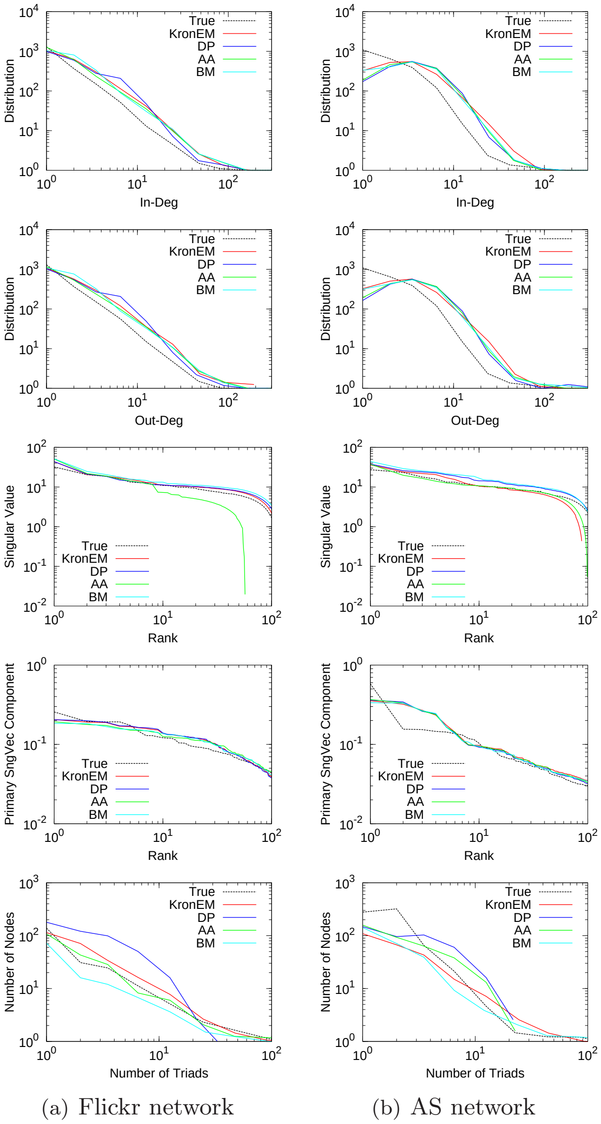
(a) In-Degree Distribution (b) Out-Degree Distribution

(c) Singular Values (d) Triad Participation

Figure 8: Network properties of the true graph H^* and the estimated properties obtained by KRONFIT and KRONEM. Notice that KRONEM almost perfectly recovers the properties of the partially observed network H^*.

B.2 Performance of KronEM in the Recovery of Network Global Properties. We already provided *PowerKS* statistics for the recovery of global network properties in Section 4.2. We achieved these statistics only considering the recovery of the missing part Z. Here we plot Figure 9 that represents the actual distribution of each network property for AS and Flickr networks. Even though it is somewhat difficult to distinguish the curves each other, KRONEM favorably performs in overall as represented in *PowerKS* statistics.

B.3 Robustness of KronEM. In Section 4.4, we briefly provided the AUC score of each method for synthetic graphs generated by various social-network models to examine the robustness of KRONEM against the underlying network structure. Here we additionally present the other scores omitted in Section 4.4: AUC_{N_Z}, LL, and LL_{N_Z}. Table 6 shows these inference performance scores on each synthetic network (Kronecker graph, Preferential attachment, Forest fire model, and Stochastic block model). These AUC_{N_Z}, LL, and LL_{N_Z} scores are in agreement not only with the corresponding AUC scores but also with the results on the real dataset, AS and Flickr networks. While the model-based approaches (BM and KRONEM) in general outperforms the classical link-prediction methods for every score (by larger than 10%), KRONEM outperforms BM in AUC and AUC_{N_Z} scores. In some LL scores, BM seems to perform better than KRONEM, however it is already shown that BM overestimates LL scores because of the large number of parameters.



(a) Flickr network (b) AS network

Figure 9: Network properties of the true Z^* and the inferred $\hat{Z}$ using KRONEM. We compare the results with the other models: Degree-Product (DP), Adamic-Adar (AA), and Stochastic block model (BM). Overall KRONEM performs favorably.

Furthermore, we also compare the performance of the recovery of global network properties on these synthetic networks as we did on the real networks. Table 7 gives the *PowerKS* statistic for each network property on the same synthetic networks as above.

Kronecker graph

Method	AUC	AUC_{N_Z}	LL	LL_{N_Z}
DP	0.749	0.5	-48,931	-7,026
AA	0.833	0.903	-52,194	-11,659
BM	0.862	0.813	-40,704	-5,993
KronEM	0.916	0.917	-38,835	-5,466

Preferential attachment

Method	AUC	AUC_{N_Z}	LL	LL_{N_Z}
DP	0.600	0.5	-131,544	-21,005
AA	0.660	0.788	-148,283	-42,715
BM	0.751	0.640	-120,721	-19,451
KronEM	0.779	0.800	-126,409	-18,959

Forest fire

Method	AUC	AUC_{N_Z}	LL	LL_{N_Z}
DP	0.865	0.5	-173,410	-26,324
AA	0.769	0.941	-211,932	-49,495
BM	0.946	0.937	-91,814	-13,677
KronEM	0.942	0.943	-117,892	-16,897

Stochastic block model

Method	AUC	AUC_{N_Z}	LL	LL_{N_Z}
DP	0.663	0.5	-82,199	-11,730
AA	0.641	0.773	-101,848	-22,089
BM	0.772	0.684	-76,000	-11,105
KronEM	0.796	0.776	-76,724	-10,338

Table 6: Inference performance (AUC) on synthetic networks generated using different network generative models (Kronecker graph (Kron), Preferential attachment (PA), Forest fire (FF) and Stochastic block model (SBM)). Performance of KronEM is stable regardless of the model used to generate the underlying synthetic network.

Kronecker graph

Method	InD	OutD	SVal	SVec	TP
DP	1.44	1.44	0.15	0.47	4.59
AA	2.16	2.05	0.22	0.28	0.59
BM	2.06	1.91	0.14	0.25	3.39
KronEM	1.94	1.77	0.22	0.29	1.11

Preferential attachment

DP	2.06	2.05	7.26	0.03	3.00
AA	1.80	1.78	0.03	0.11	1.97
BM	1.81	1.80	0.13	0.07	2.00
KronEM	1.95	1.95	0.06	0.10	2.30

Forest fire

DP	1.54	1.79	4.22	0.13	6.61
AA	2.43	1.64	0.29	0.38	3.52
BM	1.73	1.70	0.52	0.25	4.55
KronEM	1.42	1.15	0.55	0.12	4.99

Stochastic block model

DP	1.91	1.96	0.24	0.37	4.95
AA	2.05	2.18	0.15	0.37	3.39
BM	2.08	2.05	0.12	0.38	3.07
KronEM	2.43	2.45	0.44	0.45	1.19

Table 7: PowerKS statistics for synthetic graphs with different models : Kronecker graph, Preferential attachment, Forest fire, and Stochastic block model. In most networks, KronEM recovers the global network properties the most favorably.