Community Detection in Partially Observable Social Networks

Cong Tran, Won-Yong Shin, Senior Member, IEEE, and Andreas Spitz

Abstract—The discovery of community structures in social networks has gained significant attention since it is a fundamental problem in understanding the networks’ topology and functions. However, most social network data are collected from partially observable networks with both missing nodes and edges. In this paper, we address a new problem of detecting overlapping community structures in the context of such an incomplete network, where communities in the network are allowed to overlap since nodes belong to multiple communities at once. To solve this problem, we introduce KroMFac, a new framework that conducts community detection via regularized nonnegative matrix factorization (NMF) based on the Kronecker graph model. Specifically, from an inferred Kronecker generative parameter matrix, we first estimate the missing part of the network. As our major contribution to the proposed framework, to improve community detection accuracy, we then characterize and select influential nodes (which tend to have high degrees) by ranking, and add them to the existing graph. Finally, we uncover the community structures by solving the regularized NMF-aided optimization problem in terms of maximizing the likelihood of the underlying graph. Furthermore, adopting normalized mutual information (NMI), we empirically show superiority of our KroMFac approach over two baseline schemes by using both synthetic and real-world networks.

Index Terms—Community detection, influential node, Kronecker graph model, matrix factorization, overlapping community, partially observable social network

1 INTRODUCTION

1.1 Backgrounds

Real-world networks extracted from various biological, social, technological, and information systems usually contain inhomogeneities that reveal a high level of hierarchical and structural properties. Research on community detection, which is one of the most important tasks in network analysis, has thus become crucial in understanding the fundamental features (e.g., topology and functions) of these networks [1]. In general terms, communities can be regarded as the sets of points that are “close” to each other with respect to a predefined measure of distance or similarity. Since applications of community detection are diverse, there exist a variety of graph-theoretic approaches [2] that conduct optimization based on measures such as modularity [3] and conductance [4], whose performance depends heavily on network topology.

On the one hand, community detection algorithms for online social networks should be designed by taking into account their inherently overlapping and imprecise nature, since community memberships in social networks are allowed to overlap as nodes belong to multiple clusters at once [5]. The extraction of such overlapping communities is known to be more challenging than non-overlapping community detection due to a higher complexity and higher computational demands.

In practice, on the other hand, most social network data are collected from partially observable networks with both missing nodes and edges [6], which further complicates the detection of communities. For example, due to limited resources, a person or an organization may be allowed to obtain only a subset of data within a specific geographic query region. This is further compounded due to privacy settings specified by the users that may partially or entirely hide some of their traces or friendships [7]. For example, 52.6% of Facebook users in New York City hid their friend lists in June 2011 [8]. Such types of incomplete networks constitute a severe obstacle for topology-based optimization methods in detecting the true community structures. Surprisingly, while some research exists into the recovery of edges and nodes in such incomplete networks, the problem of community detection under such conditions and its solutions have not yet been investigated.

1.2 Motivation and Main Contributions

Here, we thus formulate this new problem of detecting overlapping community structures in the context of such an incomplete network in which some of the nodes and edges are
missing. To solve the problem, we present KroMFac, a new framework that intelligently combines graph recovery and community detection methods into one unified framework. To this end, KroMFac first estimates the missing part of the network using a Kronecker generative parameter matrix acquired under the Kronecker graph model. Our important contribution to the proposed framework is based on the insight that including the entirety of recovered nodes and edges in the existing graph may be detrimental to enhancement of community detection accuracy. This is because adding more recovered nodes and edges would cause the inference errors to accumulate. To address this problem, we characterize and select influential nodes by centrality ranking, which tend to have high degrees, in the effort of limiting the accumulated errors in our model. Finally, we discover the community structures by solving a regularized nonnegative matrix factorization (NMF)-aided optimization problem in terms of maximizing the likelihood of the underlying graph. Adopting normalized mutual information (NMI) as a popular information-theoretic performance metric, we empirically verify the superior performance of our proposed approach over two baselines that i) do not infer missing nodes and edges (which corresponds to the NMF method in \cite{9}) and ii) leverage recovery of the entire network. Our main contributions are four-fold and summarized as follows:

- design of a new framework, named KroMFac, that intelligently combines graph recovery and community detection in our incomplete network;
- formulation of a regularized NMF-aided joint optimization problem;
- characterization and selection of influential nodes via ranking, which play a vital role in improving community detection accuracy;
- validation of our KroMFac approach through intensive experiments based on parameter search using both synthetic and real-world datasets;
- analysis and empirical evaluation of the computational complexity.

Our framework takes an important first step towards establishing a new line of research and towards a better understanding of jointly conducting both network recovery and community detection in partially observable networks.

1.3 Organization

The remainder of this paper is organized as follows. In Section 2, we summarize significant studies that are related to our work. In Section 3, we explain the methodology of our work, including the problem definition and the overview of our KroMFac framework. Section 4 describes implementation details of the proposed KroMFac framework and two baseline schemes. Experimental results are provided in Section 5. Finally, we summarize the paper with some concluding remarks in Section 6.

1.4 Notations

Throughout this paper, the notations \( \mathbb{R} \), \( \mathbb{P}(\cdot) \), and \( \mathbb{1} \) indicate the field of real numbers, the probability, and the all-one vector, respectively; the operator \( ||\cdot||_2 \) denotes the Euclidean norm of a vector; and we use the notation \( \frac{\mathbf{A}}{\mathbf{B}} \) to denote the entry-wise division of two matrices \( \mathbf{A} \) and \( \mathbf{B} \). Unless otherwise stated, all logarithms are assumed to be to the base \( e \). Table 1 summarizes the notations used in this paper. These notations will be formally defined in the following sections when we introduce our network model and technical details.

2 Related Work

The framework that we propose in this paper is related to three broader areas of research, namely community detection in incomplete networks, detection of overlapping communities, and community detection in incomplete networks with missing edges.

**Community detection in graphs.** Since research into community detection in complex networks constitutes a very active field, there are many efforts devoted to community detection in graphs. The most popular techniques include modularity optimization \cite{3}, stochastic block models \cite{10}, spectral graph-partitioning \cite{11}, clique percolation \cite{12}, clustering \cite{13}, and label propagation \cite{14}. However, these techniques focus on graphs in which nodes can be partitioned into communities and do not address the inherent overlapping nature of community structures in many real-world networks.

**Detection of overlapping communities.** To cope with this contrast, a recently emerging topic covers the detection of overlapping communities by investigating the structural properties of such communities, especially in the case of social networks \cite{15,16,17}. As the computational complexity increases drastically for the recovery of overlapping communities instead of partitioned communities, the research has focused either on detecting communities based on local expansion \cite{18,19} or on employing scalable techniques such as NMF \cite{9,20,21} and label propagation \cite{22}.

**Community detection in incomplete networks.** Recently, research on community detection in incomplete networks with missing edges has attracted wide attention due to a lack of information caused by users’ privacy settings and limited resources. Most of these studies predict the
missing links between nodes based on the incorporated additional information [22] or the similarity of topological structures [24], [25], and then discover communities in the underlying recovered networks. In [26], a hierarchical gamma process infinite edge partition model was presented to detect communities and recover missing edges in parallel. In contrast to edge recovery, approaches for node recovery are largely still missing.

**Graph recovery in social networks.** In addition to the studies on community detection, graph recovery thus plays an important role in our research since it should precede the community detection process. A generative graph model based on Kronecker graphs, the so-called KronFit, was introduced in [27] to generate networks having structural properties of real-world networks. As the most influential follow-up study, KronEM, an approach to solving the problem of both missing nodes and edges by applying the expectation-maximization (EM) algorithm was suggested by Kim and Leskovec [28]. For cases in which only a small number of edges are missing, vertex similarity [29] was shown to be useful in recovering the original networks. Another such method for the recovery of missing edges in social networks based on shared node neighbourhoods was investigated in [30].

Despite these contributions, there has been no prior work in the literature that combines the contexts of community detection in incomplete social networks with the recovery of both missing nodes and edges. In the following, we therefore present such an approach that seamlessly integrates the recovery of missing parts of a network with subsequent community detection on the recovered network, while benefiting from a resulting more complete community structure.

## 3 Methodology

As basis for the algorithms in Section 4, we discuss network fundamentals, formalize the problem definition, and then introduce the generative graph model for graph recovery, our node selection strategies, and the community detection method.

### 3.1 Problem Definition

#### 3.1.1 Network Model and Basic Assumptions

Let us denote the partially observable network as $G = (V,E)$, where $V$ and $E$ are the set of vertices and the set of edges, respectively. The network $G$ with $N = |V|$ nodes can be interpreted as a subgraph taken from an underlying true social network $G' = (V' \cup V_M, E' \cup E_M)$, where $V_M$ is the set of unobservable nodes and $E_M$ is the set of unobservable edges. If we assume $G'$ to be a scale-free network, then the degree distribution of $G'$ can be approximated as $P(k) \sim k^{-\gamma}$, where the probability $P(k)$ of a node in the network is inversely proportional to its degree $k$ raised to the power of an exponent parameter $\gamma$. While not all real-world social networks necessarily follow a power-law distribution [32], fitting a power-law model to the long tail of the distribution is usually sufficient for practical applications. Therefore, other types of networks following a heavy-tailed degree distribution can also serve as suitable input for our work. In real-world social networks, the nodes and edges of the network $G'$ correspond to users and their relationships, respectively, with little additional information being available. In the following, we thus consider both $G$ and $G'$ to be undirected unweighted networks. Furthermore, we assume

1. The degree distribution can be estimated via least squares approximation just by taking at most 1% of the samples using a sublinear approach as indicated in [31].

### TABLE 1: Summary of notations

| Notation | Description |
|----------|-------------|
| $G$ | observable graph |
| $V$ | set of observable nodes |
| $E$ | set of observable edges |
| $V_M$ | set of missing nodes |
| $E_M$ | set of missing edges |
| $N$ | number of observable nodes |
| $M$ | number of missing nodes |
| $k$ | degree of nodes |
| $\gamma$ | degree exponent |
| $G'$ | true graph |
| $F$ | affiliation matrix |
| $C$ | number of communities |
| $A$ | adjacency matrix of the observable graph $G$ |
| $A'$ | adjacency matrix of the true graph $G'$ |
| $i$ | number of recovered nodes |
| $R^{(i)}$ | recovered graph after connecting $i$ nodes |
| $A^{(i)}_R$ | adjacency matrix of the recovered graph $R^{(i)}$ |
| $Z_1$ | matrix containing links between recovered nodes and existing nodes |
| $Z_2$ | matrix containing links between between recovered nodes |
| $H$ | number of influential nodes |
| $\lambda$ | regularization parameter |
| $\Theta$ | Kronecker parameter matrix |
| $\Theta_{init}$ | initialized Kronecker parameter matrix |
| $K$ | number of Kronecker products |
| $\text{Cen}_D$ | degree centrality |
| $\text{Cen}_K$ | Katz centrality |
| $\alpha$ | decay parameter of the Katz centrality |
| $\beta$ | constant parameter of the Katz centrality |
| $D$ | loss function |
| $\epsilon$ | threshold for determining influential nodes |
| $\psi$ | set of communities |
| $\delta$ | threshold determining communities |
| $r$ | ranking vector |
| $\eta_{select}$ | stopping criterion in node selection |
| $\eta_{detect}$ | stopping criterion in community detection |
that the number of missing nodes $M = |V_M|$ is either known or can be approximated by standard methods for estimating the size of hidden or missing populations [33]. To detect overlapping communities, we assume that social networks follow the affiliation graph model (AGM) [15], which states that the more communities a pair of nodes shares, the higher the probability that these two nodes are connected. The number of communities in the network is denoted by $C$. The AGM can be represented by a non-negative weight affiliation matrix $\mathbf{F} \in \mathbb{R}^{(N+M) \times C}$ such that each element $F_{uc}$ represents the degree of membership of a node $u \in (V \cup V_M)$ to the community $c$. The probability $P(u,v)$ of a connection between two nodes $u$ and $v$ then depends on the value of $\mathbf{F}$ and is given by $P(u,v) = 1 - \exp(-F_{uv}F_{uv}^\top)$, where $F_u \in \mathbb{R}^C$ and $F_v \in \mathbb{R}^C$ are the row vectors that correspond to nodes $u$ and $v$, respectively [9].

3.1.2 Problem Formulation

As illustrated in Fig. 1, the network structures of partially observable networks are potentially distorted significantly due to the effect of both missing nodes and edges. As a result, methods established for detecting communities may appear to perform well on the partially observable networks but are not effective in extracting the true community structures of the underlying true network.

The recovery of overlapping communities in such incomplete, partially observable networks has not been investigated before in the literature. To address this task, we thus present KroMFAC, a novel framework for recovering a partially observable network and then discovering the overlapping community structures of the recovered underlying graph. To this end, we first recover missing nodes and edges, which is equivalent to filling in the missing part of the binary adjacency matrix $\mathbf{A}' \in \{0,1\}^{(N+M) \times (N+M)}$ of the graph $G'$ based on the topological information of the observable matrix $\mathbf{A}$ (refer to Section 3.2). This inference of missing parts of the network is not without risk since adding more recovered nodes and edges may also accumulate more errors. However, many such nodes and edges may not be very relevant to the subsequent community detection. This motivates us to propose a node selection strategy that aims to characterize and include only a small number of nodes that have a high impact on the community detection. Specifically, we need to selectively recover nodes. We start by formally defining the adjacency matrix that is acquired after the iterative addition of nodes (and their edges) according to some importance ranking strategies.

Definition 1. Let $R^{(i)}$ be the selectively recovered graph formed by connecting $i \in \{0,1,\ldots,M\}$ nodes to the existing graph $G$ according to a predefined selection order. Based on the fact that $G$ and $R^{(i)}$ correspond to $\mathbf{A} \in \{0,1\}^{N \times N}$ and $\mathbf{A}_R^{(i)} \in \{0,1\}^{(N+i) \times (N+i)}$, respectively, $\mathbf{A}_R^{(i)}$ can be written as the following partitioned block matrix:

$$\mathbf{A}_R^{(i)} = \begin{bmatrix} \mathbf{A} & \mathbf{Z}_1 \\ \mathbf{Z}_1^\top & \mathbf{Z}_2 \end{bmatrix},$$

where the matrix $\mathbf{Z}_1 \in \{0,1\}^{N \times i}$ contains the links between recovered nodes and existing nodes and the matrix $\mathbf{Z}_2 \in \{0,1\}^{i \times i}$ contains the links between recovered nodes.

By definition, if we select the top $i$ nodes, then we obtain a unique matrix $\mathbf{A}_R^{(i)}$. As special cases, it follows that $A_R^{(0)} = A$ and $A_R^{(M)} = A'$. To limit the accumulated errors to a certain level in our model, we only take into account top $H \in \{0,1,\ldots,M\}$ nodes in the ranked list, termed influential nodes (refer to Definition 2 in Section 3.3).

The next step is the detection of communities, which is equivalent to estimating the affiliation matrix $\mathbf{F} \in \mathbb{R}^{(N+i) \times C}$. For each $i$, estimation of the affiliation matrix $\mathbf{F}$ leads to a probabilistic approximation of the matrix $A_R^{(i)}$ (refer to Section 3.4). Our community detection method handles two cases depending on the presence or absence of prior information on the number of communities. First, for the case where such prior information is available, when $\mathbf{F}$ has the highest chance to generate the graph $R^{(i)}$, we formulate a joint optimization problem as follows:

$$\hat{\mathbf{F}}, \hat{i} = \arg \max_{\mathbf{F} \geq 0, i \in \{0,1,\ldots,H\}} \log P(\mathbf{A}_R^{(i)} | \mathbf{F}) + \lambda \log(i+1),$$

which corresponds to a maximum log-likelihood problem with regularization for a given value of $i$. Here, $\log(i+1)$ indicates a regularization term, which is used to compensate the log-likelihood $P(\mathbf{A}_R^{(i)} | \mathbf{F})$ reduced by increasing the size of the matrix $\mathbf{A}_R^{(i)}$ since the probability $P(\mathbf{A}_R^{(i)} | \mathbf{F})$ tends to decrease with the number of elements of $\mathbf{A}_R^{(i)}$; and the parameter $\lambda > 0$ determines the impact of the regularization term and needs to be properly set according to the size of observable graphs (which will be specified in Section 5.4).

2. Here, the superscript $^*$ is used to indicate the optimal argument.

Fig. 1: An example that illustrates the difference between the community structures before and after an influential node (star) and its incident edges (black) have been deleted from the graph. Potential communities are depicted with different colors.
The role of $i$ in (1) is especially important as it is the key factor in handling the total error of the recovered graph. For the case without prior information, we formulate the following joint optimization problem:

$$\hat{F} = \arg \max_{P \geq 0} \log P(A_{R}^{|M|}|F) + \log(P(F| L)) + \log(P(L)) + \lambda \log(i + 1),$$

where $L$ represents a latent variable for the number of communities.

The overall procedure of our approach is visualized in Fig. 2.

### 3.2 Generative Graph Model

For recovery of the true network structures, the missing part of the network can be inferred by investigating the connectivity patterns in the observable part. To this end, generative models for graphs have been developed. The two major generative graph models with this aim include the stochastic block model [10] and the Kronecker graph model [34]. For our research, we adopt the Kronecker graph model since it is scalable and can be used to efficiently model a probability distribution over the missing part of social networks [34]. Thus, we briefly describe the Kronecker graph model before proceeding to network recovery.

The model is based on the Kronecker product of two graphs [35]. For two given adjacency matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{m' \times n'}$, the Kronecker product $A \otimes B \in \mathbb{R}^{mm' \times nn'}$ is defined as

$$A \otimes B = \begin{bmatrix} a_{11}B & \ldots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \ldots & a_{mn}B \end{bmatrix},$$

where $a_{uv}$ denotes the $(u, v)$th element of the matrix $A$ for $u \in \{1, \ldots, m\}$ and $v \in \{1, \ldots, n\}$. The Kronecker graph model is then defined by a Kronecker generative parameter matrix $\Theta \in [0, 1]^{N_0 \times N_0}$, where $N_0 \in \mathbb{N}$ [34]. By Kronecker-powering the parameter $\Theta$, we obtain increasingly larger and larger stochastic graph adjacency matrices. Since every entry of the matrix $\Theta$ can be interpreted as a probability, the Kronecker graph model is then equivalent to a probability distribution of edges over networks.

For network recovery, we use two algorithms, namely KronEM [28] and KronFit [27], which are built upon the Kronecker generative graph model and are the current state-of-the-art algorithms in the literature. Based on an observable network $G$, both algorithms estimate the parameter matrix $\Theta$ used to generate the full network $\Theta^K$ representing the $K$th Kronecker power of $\Theta$, where $K$ is a positive integer such that $N_0^{K-1} < N + M \leq N_0^K$. It is worth noting that KronFit infers $\Theta$ only from the observable matrix $A$ whereas KronEM leverages the existence of the missing part, i.e., $Z_1$ and $Z_2$, in inference. Let $(A_{R}^{(M)}, \sigma)$ denote a permutation matrix, where $\sigma$ indicates a permutation of the set $\{1, \ldots, N + M\}$ and $\sigma(u)$ is the index of node $u$ in the graph $R^{(M)}$ after permutation. The first $N$ elements of $\sigma$ map the nodes in $G$ while the remaining $M$ elements map the nodes in the missing part. Then, the likelihood $P(A_{R}^{(M)}, \sigma|\Theta)$ can be expressed as

$$P(A_{R}^{(M)}, \sigma|\Theta) = \prod_{a_{uv}=1}^{N} [\Theta^K]_{\sigma(u)\sigma(v)} \prod_{a_{uv}=0}^{M} (1-[\Theta^K]_{\sigma(u)\sigma(v)}),$$

where $a_{uv}$ denotes the $(u, v)$th element of the matrix $A_{R}^{(M)}$, and $[\Theta^K]_{\sigma(u)\sigma(v)}$ denotes the $(\sigma(u), \sigma(v))$th element of the matrix $\Theta^K$. As the matrix $\Theta^K$ is a probabilistic representation of $A_{R}^{(M)}$, we also obtain the missing parts $Z_1$ and $Z_2$ by assigning the value of every entry in $Z_1$ and $Z_2$ to be zero or one according to a series of Bernoulli coin-tosses with the mapped entries in $\Theta^K$ as the probabilities. The detailed steps are discussed in Section 4.

### 3.3 Influential Node Selection by Ranking

Network recovery can be seen as a statistical learning process, as we predict the value of entries in the missing part of the network by leveraging information in the observed part. Thus, after obtaining the missing part via inference, we note that using the whole recovered nodes may lead to an inaccurate detection of communities due to two types...
of errors. One type stems from the prediction model, while the other stems from random errors that occur during the Bernoulli series used to project a probabilistic value to zero or one. While the prediction error can be reducible, the random error is irreducible. For this reason, the more recovered nodes we include, the higher the sum of errors. On the other hand, using just a very small portion of missing nodes is unlikely to provide correct community structures, since there is insufficient information available to the community detection model.

Since our eventual goal is to recover the true community structures, it is intuitive to add only nodes that are useful in the community detection process. To assess the usefulness of nodes to a social network, we rely on the concept of centrality and adopt two centrality measures: degree centrality and Katz centrality [35]. Given an undirected graph, the degree centrality of node \( u \), denoted by \( \text{Cen}_D(u) \), is defined as the number of connections of a node (i.e., the number of incident edges of a node), and is computed as

\[
\text{Cen}_D(u) = \sum_{v=1}^{N+M} a_{uv}.
\]

While the degree centrality measures the number of immediate neighbors of a node, the Katz centrality also measures the influence of a node on its higher-order neighbors at larger distances. It can be seen as a generalization of the eigenvector centrality and penalizes higher-order connections with a factor \( \alpha \leq \lambda_{\text{max}}(\mathbf{A}_{R}^{(M)})^{-1} \), where \( \lambda_{\text{max}}(\mathbf{A}_{R}^{(M)}) \) is the largest eigenvector of \( \mathbf{A}_{R}^{(M)} \). The Katz centrality of node \( u \), denoted by \( \text{Cen}_K(u) \), is defined as

\[
\text{Cen}_K(u) = \alpha \sum_{v=1}^{N+M} a_{uv} \text{Cen}_K(v) + \beta
\]

with an initial condition \( \text{Cen}_K(u) = c_{\text{init}} \) for all \( u \in (V \cup V_M) \) and a constant \( c_{\text{init}} > 0 \), where \( \beta \) is a positive constant to ensure that the centrality of every node is non-zero.

To select a subset of important nodes to recover, we rank the inferred nodes by first calculating their centrality and adopt two centrality measures: degree centrality and Katz centrality.

**Definition 2.** Let \( \text{Cen}(u) \) denote a centrality measure of node \( u \in (V \cup V_M) \). Then, \( u \) is defined as an influential node if \( u \notin V \) and \( \text{Cen}(u) \geq \epsilon \), where \( \epsilon > 0 \) is a predefined threshold, \( V \) is the set of observable nodes, and \( V_M \) is the set of missing nodes. Here, \( H \in \{0, 1, \ldots, M\} \) denotes the cardinality of the set of influential nodes.

The threshold \( \epsilon \) signifies when the amount of acquired information outweighs the incurred errors from recovering parts of the true network, which means that recovering more than \( H \) nodes can be harmful to the community detection process.

### 3.4 Community Detection via Regularized NMF

Matrix factorization based approaches are commonly used tools in the detection of overlapping communities in social networks [9], [20], [21]. The benefit of these approaches lies in their scalability since many efficient techniques for solving NMF problems have been developed [37]. Additionally, the NMF-based approaches aim at detecting community structures in the whole given network in a deterministic manner, which often requires less effort to find the optimize parameter setting than those based on the label propagation. In this subsection, we describe how community detection can be transformed into two regularized NMF-aided optimization problems depending on the presence of prior information on the number of communities. Based on the theoretical considerations below, we elaborate on detailed steps for solving the combined problem of graph inference and community detection in the following section.

#### 3.4.1 Community Detection With Prior Information

First, we investigate the case where the number of communities, \( C_i \), is assumed to be known, which implies that the size of the matrix \( \mathbf{F} \) can be specified beforehand. From the previous steps, recall that we have an observation matrix \( \mathbf{A} \), a list of influential nodes, and their corresponding ranking by some centrality measures. Furthermore, all recovered matrices \( \mathbf{A}_{R}^{(i)} \) for \( i \in \{0, 1, \cdots, H\} \) are available. Due to the fact that \( \mathbb{P}(u, v) = 1 - \exp(-\mathbf{F}_u \mathbf{F}_v^\top) \) according to the AGM, the likelihood \( \mathbb{P}(\mathbf{A}_{R}^{(i)} | \mathbf{F}) \) in (1) can be rewritten as

\[
\mathbb{P}(\mathbf{A}_{R}^{(i)} | \mathbf{F}) = \prod_{a_{uv}^{(i)}=0} (1 - \exp(-\mathbf{F}_u \mathbf{F}_v^\top)) \prod_{a_{uv}^{(i)}=1} \exp(-\mathbf{F}_u \mathbf{F}_v^\top),
\]

where \( a_{uv}^{(i)} \) denotes the \((u, v)\)th element of the matrix \( \mathbf{A}_{R}^{(i)} \). Thus, we have

\[
\log(\mathbb{P}(\mathbf{A}_{R}^{(i)} | \mathbf{F})) = \sum_{a_{uv}=1} \log(1 - \exp(-\mathbf{F}_u \mathbf{F}_v^\top)) - \sum_{a_{uv}=0} \exp(-\mathbf{F}_u \mathbf{F}_v^\top).
\]

Suppose that \( f(\mathbf{X}) = 1 - \exp(-\mathbf{X}) \) for a matrix \( \mathbf{X} \). Then, we obtain a matrix \( f(\mathbf{FF}^\top) \) that probabilistically approximates the adjacency matrix \( \mathbf{A}_{R}^{(i)} \). To estimate the difference between two matrices \( \mathbf{A}_{R}^{(i)} \) and \( f(\mathbf{FF}^\top) \), instead of using the Euclidean distance metric, we utilize the negative log-likelihood in (5) as a loss function \( D \), which indicates that

\[
D(\mathbf{A}_{R}^{(i)} , f(\mathbf{FF}^\top)) = -\log(\mathbb{P}(\mathbf{A}_{R}^{(i)} | \mathbf{F})).
\]

As a result, the optimization problem in (1) can then be cast into a regularized NMF formulation as

\[
(\hat{\mathbf{F}}, \hat{i}) = \arg \min_{\mathbf{F} \geq 0, i \in \{0, 1, \cdots, H\}} D(\mathbf{A}_{R}^{(i)} , f(\mathbf{FF}^\top)) - \lambda \log(i+1),
\]
where the objective function in (7) is referred to as the regularized loss in our setup.

### 3.4.2 Community Detection Without Prior Information

To detect community structures without prior information on \( G \), we adopt Bayesian NMF (BNMF) \([20]\), where a scalable hyperparameter matrix \( \Theta \in \mathbb{R}^{(N+1) \times (N+1)} \) is used to represent a latent variable for \( C \). As we aim to maximize the posterior \( P(F, L | A_R^{(i)}) \), the loss function \( D \) in (5) is rewritten as

\[
D(A_R^{(i)}, f(FF^\top)) = -\log(P(A_R^{(i)} | F)) - \log(P(F | L)) - \log(P(L)).
\]

Using (8), we then obtain a regularized NMF formulation as in (7).

### 4 Proposed KroMFac Framework

In this section, to provide a complete solution to the problem of community detection in a partially observable graph, we present KroMFac, a novel framework that consists of the following three major phases: 1) graph recovery, 2) node ranking and selection, and 3) community detection. The overall procedure is described in Algorithm 1. The observable graph \( G \), the number of missing nodes, \( M \), and the number of communities, \( C \) (if it is known), are the key input parameters of the algorithm. The dimension of the parameter matrix \( \Theta \) is given by \( N_0 \times N_0 \), and we initialize \( \Theta \) as a randomly generated matrix \( \Theta_{\text{init}} \in [0, 1]^{N_0 \times N_0} \). Further parameters serve as control parameters. In particular, \( \alpha, \beta, \) and \( \epsilon \) play a central role in determining the set of influential nodes and are introduced in detail in Section 3.3 and \( \lambda \) controls the impact of regularization, which can be quantified via an empirical study. The parameter \( \delta > 0 \) serves as a threshold to decide to which communities each node belongs (i.e., the degree of membership of nodes), and can be estimated for a given network \([9]\). Finally, parameters \( \eta_{\text{select}} \) and \( \eta_{\text{detect}} \) are arbitrarily small positive constants used as stopping criteria during node selection and community detection (i.e., convergence criteria). As the output of Algorithm 1, we define \( \psi \) as the set of detected communities.

We assume that all communities initially have no members. To find the true community structures of the incomplete input graph, we first fully recover the graph (refer to Algorithm 2). By analyzing the recovered graph, we then select \( H \) influential nodes and determine the ranking vector \( r \in \mathbb{N}^H \) that represents the indices of ranked influential nodes and plays a crucial role in community detection accuracy (refer to Algorithm 3). In this step, specifically, we solve the joint optimization problem described in (7) through exhaustive search over \( i \) by sequentially connecting influential nodes to the existing observable graph based on the order in \( r \). For each \( i \), we acquire a corresponded graph \( R^{(i)} \) and its adjacency matrix \( A_R^{(i)} \) (see Definition 1). By using either Algorithm 4 or 5, we are then capable of obtaining the loss function \( D \) associated with affiliation matrix \( F \) given the input matrix \( A_R^{(i)} \). As a result, we obtain \( D_{\min} \) as the smallest value of \( D - \lambda \log(i + 1) \) (i.e., the regularized loss), which in turn provides us with the optimal \( F \) and \( i \). Every entry \( F_{uc} \) in the optimal affiliation matrix \( F \) then describes the likelihood that the node \( u \in (V \cup V_M) \) belongs to community \( c \in \{1, \ldots, C\} \). Therefore, it is possible to recover the community structures \( \psi \) by assigning node \( u \) to community \( c \) if the corresponding entry \( F_{uc} \) is greater than or equal to the threshold \( \delta \). In the following, we elaborate on each major phase of the KroMFac framework.

#### 4.1 Graph Recovery

The first step of our framework is the inference of missing parts of the graph from priors on the observable matrix \( A \) and the number of missing nodes, \( M \), by using the function GraphRecv in Algorithm 2, where either the function KronEM or KronFit is employed. First, by applying the KronEM algorithm, from an initialized \( \Theta_{\text{init}} \), the E-step samples the missing parts \( Z_1 \) and \( Z_2 \), and the permutation

---

**Algorithm 1: KroMFac**

**Input:** \( G, M, C, N_0, \Theta_{\text{init}}, \alpha, \beta, \epsilon, \delta, \eta_{\text{select}}, \eta_{\text{detect}}, \lambda \)

**Output:** \( \psi \)

1. **Initialization:** \( D_{\min} \leftarrow \infty; \psi[c] \leftarrow \{\emptyset\} \) for \( c \in \{1, \ldots, C\} \)

2. **function KroMFac**

   1. \( A \leftarrow \) Adjacency matrix of \( G \)
   2. \( A_R^{(M)} \leftarrow \) GraphRecv(\( A, N_0, \Theta_{\text{init}}, M \))
   3. \( (H, r) \leftarrow \) NodeSelect(\( A_R^{(M)}, M, \alpha, \beta, \epsilon, \eta_{\text{select}} \))
   4. **for** \( i \) from 1 to \( H \) **do**
   5. \( R^{(i)} \leftarrow \) Connect nodes \( \{r[1], \ldots, r[i]\} \) to \( G \)
   6. \( A_R^{(i)} \leftarrow \) Adjacency matrix of \( R^{(i)} \)
   7. **if** \( C \) is known, **then**
   8. \( (D, F) \leftarrow \) BIGCLAM(\( A_R^{(i)}, C, \eta_{\text{detect}} \))
   9. **else**
   10. \( (D, F) \leftarrow \) BNMF(\( A_R^{(i)}, \eta_{\text{detect}} \))
   11. \( D \leftarrow D - \lambda \log(i + 1) \)
   12. **if** \( D_{\min} < D \) **then**
   13. \( F \leftarrow F \)
   14. \( i \leftarrow i \)
   15. \( D_{\min} \leftarrow D \)
   16. **for** \( u \) from 1 to \( N + M \) **do**
   17. **for** \( c \) from 1 to \( C \) **do**
   18. **if** \( F_{uc} \geq \delta \) **then**
   19. \( \psi[c] \leftarrow \psi[c] \cup \{u\} \)
   20. **return** \( \psi \)
σ. In the M-step, a stochastic gradient descent process subsequently optimizes the parameter matrix Θ given the samples obtained in the E-step. The EM iteration alternates between performing the E-step and M-step according to the following expressions, respectively:

**E-step:**
\[
(Z_1^{(t)}, Z_2^{(t)}, σ^{(t)}) \sim F(Z_1, Z_2, σ | A, Θ^{(t)}),
\]

**M-step:**
\[
Θ^{(t+1)} = \arg \max_{Θ ∈ (0,1)^{N_0}} E[F(Z_1^{(t)}, Z_2^{(t)}, σ^{(t)}, A | Θ)],
\]

where the superscript (t) denotes the iteration index. On the other hand, the KronFit algorithm can be regarded as a single M-step in the above EM algorithm, where we do not generate the missing parts \(Z_1^{(t)}\) and \(Z_2^{(t)}\). As a result, both KronEM and KronFit return \(Θ = Θ^{(t)}\) and \(σ = σ^{(t)}\) (see lines 5–8 in Algorithm 2).

Then, we generate the stochastic adjacency matrix \(Θ^K\).

To create the fully recovered matrix \(A_R^{(M)}\), for the first \(N\) rows and columns of \(A_R^{(M)}\), we replicate the entries of matrix \(A\) in the upper left of matrix \(A_R^{(M)}\) (refer to lines 12–13 in the algorithm). To infer the missing part (i.e., the last \(M\) rows and columns of \(A_R^{(M)}\)), we consecutively run the Bernoulli trials with the probability \(Θ^K_{σ(u)σ(v)}\) and then map the value of the missing entry in row \(u\) and column \(v\) to one if a success occurs and zero otherwise. Since the adjacency matrix \(A_R^{(M)}\) is symmetric, we only need to repeat this process \(M N + \frac{M^2}{2}\) times. An example of this graph recovery phase is illustrated in Fig. 3 when \(N = 6, M = 2, N_0 = 2, K = 3, and σ(u) = u\) for \(u \in (V ∪ V_M)\). In this example, since the last two rows and columns of \(Θ^3\) correspond to two recovered nodes, we execute Bernoulli trials on each non-zero entry in this part to obtain the recovered matrix \(A_R^{(M)}\). If the number of missing edges can be estimated (see [28] for details), then the termination of this final step can be accelerated for sparse graphs since the Bernoulli trials can be terminated once the number of entries with a value that is equal to one exceeds the predicted number of edges.

### 4.2 Node Ranking and Selection

Based on the output of the GraphRecv algorithm (i.e., the recovered matrix \(A_R^{(M)}\)), the function NodeSelect ranks missing nodes and then selects influential nodes from the set of ranked candidates. Conceptually, numerous centrality measures could be applied to obtain such a ranking. In our work, we focus on two well-known centrality measures for ranking nodes, namely degree centrality \(Cen_D(u)\) and Katz centrality \(Cen_K(u)\) for node \(u \in (V ∪ V_M)\). As the degree centrality in [3] is rather straightforward to be computed, in Algorithm 3, we only present node selection based on Katz centrality (see [4] with input parameters \(α\) and \(β\) (refer to lines 3–7 in the algorithm, where both \(Cen\) and \(Cen_{temp}\) are \((N + M)\)-dimensional row vectors used to store the centrality values of missing nodes, and \(0\) and \(1\) are all-zero and all-one row vectors, respectively). In our algorithm, the convergence parameter \(η_{select} > 0\) can be set to an arbitrarily small value, and a reasonable value of \(ε\) will be specified.
via numerical search in the next section. To ensure that the Katz centrality converges, it has been shown in [38] that parameters $\alpha$ and $\beta$ can be set as in the following:
\begin{align}
\alpha &= \lambda_{\max}(A^{(M)}_R)^{-1}, \\
\beta &= 1,
\end{align}
respectively, where $\lambda_{\max}(A^{(M)}_R)$ is the largest eigenvalue of $A^{(M)}_R$. Next, we introduce the ranking vector $r \in \mathbb{N}^M$ to record the centrality ranking of missing nodes. Since we aim to select nodes whose centrality measures are greater than a given threshold $\epsilon$ (refer to Section 3.3), only $H \leq M$ most influential nodes are associated with $r$. For example, $r[u]$ represents the index of the node ranked at the $u$th position in the list. Algorithm 3 returns the number of influential nodes, $H$, and the ranking vector $r \in \mathbb{N}^H$. Figure 4 shows a simple illustration of this node ranking and selection phase when $\epsilon = 2, M = 2$, and $\text{Cen}(u) = \text{Cen}_D(u)$. In this figure, due to the fact that the last two rows and columns in the input matrix $A^{(M)}_R$ correspond to two recovered nodes, we calculate the degree centrality of the two nodes and select the seventh placed node as an influential node since its centrality is greater than or equal to $\epsilon$.

4.3 Community Detection

In this subsection, we describe two types of community detection algorithms depending on the presence or absence of prior information on the number of communities, $C$.

4.3.1 Community Detection With Prior Information

To solve the problem of community detection with known $C$, we use the function BIGCLAM [9] in Algorithm 4, which is the state-of-the-art NMF-aided detection method. For given $i$, we solve (7) using a block coordinate gradient ascent algorithm [37]. In particular, if we update a row vector $F_u$ of the affiliation matrix $F$ with other fixed rows $F_v$, then the problem of updating $F_u$ becomes convex, which can be easily solved using the projected gradient ascent method along with the gradient $\nabla l(F_u)$ given by line 5 in Algorithm 4. The step size $\Delta u > 0$ is computed via backtracking line search [39]. After the update process, we project $F_u$ onto a space of nonnegative vectors by setting $F_{uc} = \max(F_{uc}, 0)$. The optimization process is terminated when the change in each iteration, denoted by $\Delta D > 0$, is less than an arbitrarily small threshold $\eta_{\text{select}} > 0$. The algorithm returns the loss function $D$ and the corresponding matrix $F$.

4.3.2 Community Detection Without Prior Information

The function BNMF [20] in Algorithm 5 adopts a Bayesian NMF-aided approach to solving the problem of community detection without prior information on $C$ for given $A^{(i)}$. In this algorithm, the hyperparameter $L$ is input in the form of a diagonal matrix whose $(k,k)$th element is denoted by $L_{kk}$. $L$ is initially set to 1 for all $k \in \{1, \cdots, N+i\}$ and follows a Gamma distribution with two input hyper-hyperparameters $a$ and $b$, which are randomly generated. In Algorithm 5, due to a lack of information on $C$, we start by randomly initializing a matrix $\hat{F} \in \mathbb{R}^{(N+i) \times (N+i)}$, which corresponds to the affiliation matrix, where $\hat{F}_k$ denotes the $k$th row vector of $\hat{F}$. By using a fast fixed-point algorithm [40], we update $\hat{F}$ and $L$ until the change in each iteration, $\Delta D > 0$, is less than a
Algorithm 4: BIGCLAM [9]

Input: \( A^{(i)}_R, C, \eta_{\text{detect}} \)
Output: \( D, F \)
1 Initialization: \( F \leftarrow \text{randomly initialize}; D_{\text{prev}} \leftarrow \infty \)
2 function BIGCLAM
3 while \( \Delta D \geq \eta_{\text{detect}} \) do
4 for \( u \) from 1 to \( N + i \) do
5 \( \nabla l(F_u) \leftarrow \sum_{v \in N(u)} F_v \log(1 - \exp(-F_u F_v^\top)) - \sum_{v \notin N(u)} F_v F_v^\top \)
6 Compute step size \( \Delta_s \)
7 for \( c \) from 1 to \( C \) do
8 \( F_{uc} \leftarrow F_{uc} + \Delta_s \nabla l(F_u) \)
9 \( F_{uc} \leftarrow \max(F_{uc}, 0) \)
10 \( D \leftarrow \sum_{a,v=0} (F_u F_v^\top) - \sum_{a,v=1} \log(1 - \exp(-F_u F_v^\top)) \)
11 \( \Delta D \leftarrow |D - D_{\text{prev}}| \)
12 \( D_{\text{prev}} \leftarrow D \)
13 return \( (D, F) \)

Algorithm 5: BNMF [20]

Input: \( A^{(i)}_R, \eta_{\text{detect}} \)
Output: \( D, F \)
1 Initialization: \( \tilde{F} \leftarrow \text{randomly initialize}; D_{\text{prev}} \leftarrow \infty \); \( a \leftarrow \text{randomly initialize}; b \leftarrow \text{randomly initialize} \)
2 function BNMF
3 while \( \Delta D \geq \eta_{\text{detect}} \) do
4 \( \tilde{F} \leftarrow \left( \frac{F}{\|F\|_F} \right) \left( \frac{A^{(i)}_R}{\|F\|_F} \right) \tilde{F} \)
5 for \( k \) from 1 to \( (N + i) \) do
6 \( L_{kk} \leftarrow \frac{A^{(i)}_R}{\|F\|_F^2 + b} \)
7 \( \nabla^2 l(F) = \frac{\partial^2 l(A_R)}{\partial F^2} \)
8 \( \nabla^2 l(F) = -\log(P(A_R|F)) - \log(\tilde{F}|L) - \log(L) \)
9 \( \Delta D \leftarrow |D - D_{\text{prev}}| \)
10 return \( (D, F) \)

Algorithm 5: KroMFac

Input: \( A^{(i)}_R, \eta_{\text{detect}} \)
Output: \( D, F \)
1 Initialization: \( \tilde{F} \leftarrow \text{randomly initialize}; D_{\text{prev}} \leftarrow \infty \); \( a \leftarrow \text{randomly initialize}; b \leftarrow \text{randomly initialize} \)
2 function KroMFac
3 while \( \Delta D \geq \eta_{\text{detect}} \) do
4 \( \tilde{F} \leftarrow \left( \frac{F}{\|F\|_F} \right) \left( \frac{A^{(i)}_R}{\|F\|_F} \right) \tilde{F} \)
5 for \( k \) from 1 to \( (N + i) \) do
6 \( L_{kk} \leftarrow \frac{A^{(i)}_R}{\|F\|_F^2 + b} \)
7 \( \nabla^2 l(F) = \frac{\partial^2 l(A_R)}{\partial F^2} \)
8 \( \nabla^2 l(F) = -\log(P(A_R|F)) - \log(\tilde{F}|L) - \log(L) \)
9 \( \Delta D \leftarrow |D - D_{\text{prev}}| \)
10 return \( (D, F) \)

4.4 Analysis of Computational Complexity

In this subsection, we analyze the computational complexity of the KroMFac framework. Since our framework consists of three major phases including network recovery, influential node selection, and community detection, we elaborate on the complexity analysis of each phase. We show the complexity only for the case of KronEM–BIGCLAM and degree centrality, which leads to the best performance among all combinations (see Section 5.4.1 for more details). To reduce the complexity, we take advantage of the property that real-world social networks usually have a sparse and low-rank matrix structure [41].

The graph recovery phase is composed of the KronEM algorithm and the Bernoulli trials. The KronEM algorithm was shown in [28] to have the complexity of \( O(|E| \log |E|) \), where \( |E| \) is the number of edges in the partially observable network \( G \). Since the Bernoulli trials have a complexity of \( O(|E|) \) in sparse graphs, the complexity of graph recovery is bounded by \( O(|E| \log |E|) \). In the node selection phase, the computation of degree centrality dominates the complexity, which is given by \( O(|E|) \) in sparse graphs. In the community detection phase, an almost linear complexity in \( N \) can be achieved via the approach in [9], where \( N \) denotes the number of observable nodes. Here, while the community detection process is repeated \( H \) times, one can see that \( H \ll M \) (see Section 5.4.1), where \( H \) and \( M \) denote the numbers of influential nodes and missing nodes, respectively. From the fact that \( M \) is smaller than \( N \), we can deduce that the complexity of this phase is bounded by \( O(N) \). Hence, the total computational complexity of KroMFac is finally given by \( O(|E| \log |E|) \).

5 EXPERIMENTAL EVALUATION

In this section, we first describe both synthetic and real-world datasets. We also present two baseline schemes for community detection as a comparison. By adopting the NMI as a popular information-theoretic performance metric, we then present the performance of our community detection framework and compare it against the two baseline schemes. Additionally, we investigate the complexity of our KroMFac framework.

5.1 Datasets

To evaluate the community detection performance of our approach, we rely on datasets for which ground-truth communities are explicitly labeled. In the following, both synthetic and real-world datasets across various domains are taken into account.

5.1.1 Synthetic Datasets

We construct synthetic graphs via the extended Lancichinetti-Fortunato-Radicchi (LFR) benchmark [42], which is built upon a generative model that creates nodes along with prior known community labels. The benchmark is capable of generating graphs that replicate important features of real social networks such as the power-law...
degree distribution and overlapping communities. To create an LFR graph, ten parameters need to be specified, which are summarized in Table 2. While parameters such as the number of nodes, average degree, maximum degree, maximum and minimum community size, and degree exponent are rather straightforward to understand, we explain the remaining four parameters:

- The community size exponent refers to the exponent parameter from a power-law approximation of the distribution of the number of nodes in communities.
- The mixing parameter, denoted by \( \mu \), controls the proportion of random edges to total edges; for example, if \( \mu = 0.3 \), then the LFR benchmark produces a graph such that approximately 70% of edges link to nodes within the same community, while the remaining 30% connect to nodes in other randomly selected communities. This parameter is sensitive to the performance of community detection. In general, if \( \mu \) is closer to one, then the community structures become weaker. On the other hand, when \( \mu \) is closer to zero, one can expect high detection performance since community structures can be easily identified.
- The number of overlapping nodes refers to the number of nodes in the graph that belong to more than one community.
- The number of communities per node indicates the average number of communities to which each of the overlapping nodes belongs.

To cover various domains of network applications, we generate three LFR graphs with differing parameter settings as specified in Table 2.

### 5.1.2 Real-World Datasets

To validate the applicability of our approach, three types of real-world datasets are also used for evaluation. More specifically, from the available SNAP datasets [43], we use the Amazon product co-purchasing network [44], the collaboration network of DBLP [45], and the Youtube videosharing social network [46]. The statistics of these datasets are summarized in Table 3 and the basic characteristics of each network are described in the following:

- **The number of nodes (N.N.):** In the Amazon network, nodes represent products. In DBLP, nodes represent authors. In the Youtube network, nodes represent users.
- **The number of edges (N.E.):** In the Amazon network, edges connect products that are commonly purchased together. In DBLP, two authors are connected by an edge if they have co-authored a paper. In the Youtube network, edges represent friendships between users.
- **The number of communities (N.C.):** In the Amazon network, each product category corresponds to a ground-truth community. In DBLP, the publication venues are used as ground-truth communities. In the Youtube network, user-created groups are used as ground-truth communities.
- **Average community size (A.C.S.):** The average number of nodes within communities.
- **Community memberships per node (C.M.N.):** The average number of communities that a node belongs to.

### 5.2 Baseline Approaches

Due to the fact that community discovery in partially observable networks with both missing nodes and edges has never been studied in the literature, there is no state-of-the-art method that works appropriately under our network model. For this reason, we present our own two types of baseline schemes by taking into account some special cases of our KroMFac framework.

#### 5.2.1 Baseline 1 (Community Detection)

As a naive approach, the first baseline scheme (Baseline 1) aims to directly discover community structures based on an observable network via the NMF-aided detection method without recovering any nodes and edges. To this end, Baseline 1 solves an optimization problem such that the matrix \( \mathbf{F} \) is found given an adjacency matrix \( \mathbf{A} \) of the incomplete network. The problem formulations with and without prior information on the number of communities, \( C \), are thus given by

\[
\mathbf{F} = \arg \max_{\mathbf{F}} \mathbb{P}(\mathbf{A}|\mathbf{F})
\]

\[
\mathbf{F} = \arg \max_{\mathbf{F} \geq 0} \mathbb{P}(\mathbf{A}|\mathbf{F})
\]
and
\[
\hat{F} = \arg\max_{F \geq 0} \log \mathbb{P}(A|F) + \log(\mathbb{P}(F|L)) + \log(\mathbb{P}(L)),
\]
respectively, where \(L\) represents a latent variable for \(C\). This corresponds to a special case with no regularization term where \(i\) is set to zero in our joint optimization problem \([1]\). Similarly as in the methodology in Section 5.4, the optimal \(F\) can be easily acquired via Algorithm 4 or 5 by replacing the input matrix \(A^{(M)}\) by \(A\). As both algorithms result in \(\hat{F}\) providing fuzzy information on the community memberships, we apply the hard-decision process (refer to lines 18–21 in Algorithm 1) to find the community structure \(\psi\).

5.2.2 Baseline 2 (Graph Recovery + Community Detection)
In addition to Baseline 1, to highlight the importance of node ranking and selection, we also present the second baseline scheme (Baseline 2) that performs community detection along with a fully recovered graph. To this end, Baseline 2 solves an optimization problem such that the matrix \(A\) is found given an adjacency matrix \(A^{(M)}\) that can be first inferred by the graph recovery phase. The problem formulations with and without prior information on \(C\) are thus given by
\[
\hat{F} = \arg\max_{F \geq 0} \mathbb{P}(A^{(M)}|F)
\]
and
\[
\hat{F} = \arg\max_{F \geq 0} \log \mathbb{P}(A^{(M)}|F) + \log(\mathbb{P}(F|L)) + \log(\mathbb{P}(L)),
\]
respectively. This corresponds to another special case with no regularization term, where \(i\) is set to \(M\) in \([1]\). Note that node ranking is not necessary since all recovered nodes are inserted into the graph. To solve this problem, we first follow the steps similar to those in Section 5.2 to recover the matrix \(A^{(M)}\). After the graph recovery phase in Algorithm 2, the optimal \(F\) can be acquired via either Algorithm 4 or 5 by assuming that \(i = M\). Then, the hard-decision process in Algorithm 1 is performed to produce the final result \(\psi\). In the next section, we experimentally show that the above scheme that does not employ the node ranking and selection phase (i.e., Baseline 2) significantly degrades the detection performance, compared to our original KroMFac framework.

5.3 Performance Metric
To assess the performance of our KroMFac framework and two baseline schemes, we need to quantify the degree of agreement between the ground-truth communities and the detected communities. In particular, given a set of true labels and the set of labels assigned by the resulting community detection, we need to find the similarity between them. While there are various ways to estimate the similarity, the NMI is one of the most widely used evaluation measures for community detection problems \([9]\), \([20]\), \([47]\), and is formally defined as in the following.

**Definition 3 (NMI [48])** Assume that the community assignments are \(x_i\) and \(y_i\), where \(x_i\) and \(y_i\) indicate the labels of vertex \(i\) in the true community \(X\) and the predicted community \(Y\), respectively. When the labels \(x\) and \(y\) are the values of two random variables \(X\) and \(Y\), following a joint distribution \(P(x,y) = P(X = x, Y = y)\), the NMI between \(X\) and \(Y\) is given by
\[
\text{NMI}(X,Y) = 1 - \frac{1}{2} \left( \frac{H(X|Y)}{H(X)} + \frac{H(Y|X)}{H(Y)} \right),
\]
where \(H(X) = \sum_x P(x) \log P(x)\) is the Shannon entropy \([49]\) of \(X\) and \(H(X|Y) = \sum_{x,y} P(x,y) \log P(x|y)\) is the conditional entropy of \(X\) given \(Y\).

5.4 Experimental Results
In our experiments, for both synthetic and real-world datasets, we use the following parameter settings: \(\alpha\) and \(\beta\) are set according to \([9]\) and \([10]\), respectively; both \(\eta_{\text{select}}\) and \(\eta_{\text{detect}}\) are set to \(10^{-4}\); and the regularization parameter \(\lambda\) is set to \(10N\), where \(N\) is the number of observable nodes. To create partially observable networks from the original graphs, we adopt two graph sampling strategies from \([50]\).

The first strategy, called random node (RN) sampling, selects nodes uniformly at random to create a sample graph. The second one, forest fire (FF) sampling, starts by picking a seed node uniformly at random and adding it to a sample graph (referred to as burning). Then, FF sampling burns a fraction of the outgoing links with nodes attached to them. This process is recursively repeated for each neighbor that is burned until no new node is selected to be burned. Both sampling strategies are known not to be biased towards high degree nodes, while FF sampling is capable of preserving the degree distribution of the original graph \([50]\).

5.4.1 Synthetic Datasets
We create partially observable networks consisting of 7,000 nodes from the original three synthetic datasets mentioned in Section 5.1.1 by performing two aforementioned graph sampling strategies. To be consistent in evaluating the performance of community detection in the incomplete graphs, we also perform node deletion such that the ground-truth community structures \(\psi\) do not contain nodes removed from the original graphs.

For ease of illustration, we show only the results obtained for the combination of KronEM–BIGCLAM and Graph 2 in Figs. 5 and 6 since other cases follow similar...
trends. First, we compare the performance of degree centrality and Katz centrality for measuring node importance in our approach. In Fig. 5, we show the NMI against the number of recovered nodes, $i$, when communities are detected for given $i$ nodes that are ranked based on both the degree centrality and the Katz centrality. From Fig. 5a, we observe that the maximum NMI is given by 0.1597 and 0.1527 at $i = 15$ and $i = 14$ when the degree centrality and Katz centrality measures are employed, respectively. A similar observation can be made in Fig. 5b. Clearly, recovered nodes ranked by the degree centrality provide better NMI performance for both sampling strategies, and we thus adopt this centrality measure for our subsequent experiments. Second, for our KroMFac framework, we investigate how close the optimal $i$ that is the solution to (7) is to the value of $i$ that maximizes the NMI. In Fig. 6a, we illustrate the regularized loss (i.e., the objective function in (7)) over $i$ when FF sampling is applied, where the minimum loss is attained at $i = 24$ at which the minimum loss is attained. (b) NMI. Here, the black circle depicts the point $i = 24$ at which the maximum NMI is achieved.

Finally, the NMI of KroMFac and two baseline schemes for all three synthetic graphs is shown in Table 5 when both RN and FF sampling strategies are applied. We find that our KroMFac framework significantly outperforms the baselines for all synthetic graphs with improvement rates of up to 17.15% and 21.28% over Baselines 1 and 2, respectively. From the table, it is also clear that the NMI performance of Baseline 2 is almost comparable to or even worse than that of Baseline 1, which shows that including the entirety of recovered nodes and edges is not beneficial.

5.4.2 Real-World Datasets

We create partially observable networks by deleting some nodes and all their associated edges from the original real-world datasets through two graph sampling strategies. More specifically, we delete $150 \times 10^3$ nodes from the original Amazon and DBLP datasets and $300 \times 10^3$ nodes from the original Youtube dataset. We also delete the corresponding nodes from the ground-truth community structures $\psi$. Using the degree centrality and $\epsilon = \frac{k_{\text{max}}}{2}$ as determined above, we show the NMI of KroMFac and two baseline

| Method          | RN sampling | FF sampling |
|-----------------|-------------|-------------|
| KronEM–BIGCLAM  | 0.1585      | 0.2141      |
| KronEM–BNMF     | 0.1374      | 0.1945      |
| KronFit–BIGCLAM | 0.1348      | 0.1935      |
| KronFit–BNMF    | 0.1245      | 0.1871      |
TABLE 5: NMI of KroMFac and two baseline schemes for three synthetic graphs

| NMI   | KroMFac (X) | Baseline 1 (Y) | Baseline 2 (Z) | Improvement rate (%) |
|-------|-------------|----------------|----------------|----------------------|
| Graph 1 (RN) | 0.2485 | 0.2205 | 0.2049 | 12.70 21.28 |
| Graph 1 (FF) | 0.3996 | 0.3602 | 0.3485 | 10.94 14.66 |
| Graph 2 (RN) | 0.1585 | 0.1353 | 0.1346 | 17.15 17.76 |
| Graph 2 (FF) | 0.2141 | 0.1929 | 0.1779 | 10.99 20.35 |
| Graph 3 (RN) | 0.1412 | 0.1212 | 0.1165 | 16.50 21.20 |
| Graph 3 (FF) | 0.2313 | 0.2032 | 0.1988 | 13.83 16.35 |

TABLE 6: NMI of KroMFac and two baseline schemes for three real-world datasets

| NMI   | KroMFac (X) | Baseline 1 (Y) | Baseline 2 (Z) | Improvement rate (%) |
|-------|-------------|----------------|----------------|----------------------|
| Amazon (RN) | 0.1448 | 0.1327 | 0.1219 | 9.14 18.78 |
| Amazon (FF) | 0.1962 | 0.1837 | 0.1727 | 6.79 13.59 |
| DBLP (RN) | 0.1667 | 0.1436 | 0.1399 | 16.11 19.15 |
| DBLP (FF) | 0.2866 | 0.2605 | 0.2583 | 10.02 10.97 |
| Youtube (RN) | 0.1263 | 0.1179 | 0.1058 | 7.12 19.34 |
| Youtube (FF) | 0.1556 | 0.1514 | 0.1431 | 2.80 8.73 |

Schemes for two real-world graphs in Table 6. It is clear from the results that our KroMFac framework is superior to the baselines for two real-world networks with improvement rates of up to 16.11% and 19.34% over Baselines 1 and 2, respectively. From Table 6, it is also clear that the NMI performance of Baseline 2 is even slightly inferior to that of Baseline 1.

5.4.3 Empirical Evaluation of Complexity

We empirically show the average runtime complexity via experiments using synthetic graphs sampled by FF from Graph 2 for different numbers of sampled nodes with \( N + M = 2^k \) and \( k \in \{7, \cdots, 13\} \). Then, 30% of nodes and their associated edges are deleted by FF sampling to create partially observable networks. Other parameters of the LFR benchmark are set proportionally to the number of nodes (refer to Table 2). Parameters of the KroMFac framework follow the same settings as in Section 5.4.1. In Fig. 7, we illustrate the plot of the runtime complexity in seconds versus \( |E| \), where each point is the average of experimental results obtained by executing the KroMFac process 10 times. An asymptotic curve \( |E| \log |E| \) is also shown in the figure, where it manifests trends consistent with our experimental results.

Moreover, we note that since the computation based on a large-size matrix is expensive in terms of memory consumption, it is necessary to adopt cost-effective techniques (e.g., coordinate format) to store and compute sparse matrices.

6 CONCLUDING REMARKS

In this paper, the problem of discovering overlapping community structures in the context of partially observable networks with both missing nodes and edges was introduced for the first time. To solve this problem, a novel framework, termed KroMFac, that seamlessly incorporates graph recovery into community recovery was developed. Specifically, community detection was performed via regularized NMF based on the Kronecker graph model. In particular, motivated by the insight that adding a proper number of missing nodes and edges to the existing graph would be of significant importance in improving community detection accuracy, we presented how to characterize and select influential nodes via centrality ranking. By adopting the
NMI as a performance metric, our KroMFac framework was validated through experiments on both synthetic and real-world datasets. Based on parameter search, we showed that our approach outperforms two baselines by a large margin on synthetic and real-world networks. Additionally, we analytically and empirically examined the computational complexity of our framework.

Potential avenues of future research in this area include analyzing a fundamental limit of our solutions to the problem of both graph recovery and community detection as a form of error bound. Another interesting direction is the inclusion of deep generative graph models for community detection to reduce the inference error even further.

ACKNOWLEDGMENTS

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2017R1D1A1A09000835). Won-Yong Shin is the corresponding author.

REFERENCES

[1] S. Fortunato, “Community detection in graphs,” Phys. Rep., vol. 486, no. 3, pp. 75–174, Feb. 2010.
[2] J. Leskovec, K. J. Lang, and M. Mahoney, “Empirical comparison of algorithms for network community detection,” in Proc. 19th Int. Conf. World Wide Web (WWW), Rayleigh, NC, Apr. 2010, pp. 631–640.
[3] M. E. Newman, “Modularity and community structure in networks,” Proc. of the National Acad. Sci., vol. 103, no. 23, pp. 8577–8582, Apr. 2006.
[4] Y. Zhang and D.-Y. Yeung, “Overlapping community detection via bounded nonnegative matrix tri-factorization,” in Proc. 18th Int. Conf. Knowl. Disc. and Data Mining (SIGKDD), Beijing, China, Aug. 2012, pp. 606–614.
[5] B. Yan and S. Gregory, “Finding missing edges and communities in incomplete networks,” J. Phys. A: Math. and Theor., vol. 44, no. 49, p. 495102, Nov. 2011.
[6] R. Dey, Z. Jelveh, and K. Ross, “Facebook users have become much more private: A large-scale study,” in Proc. IEEE Int. Conf. Pervasive Comput. Commun. Worksh. (PERCOM), Lugano, Switzerland, 2012, pp. 346–352.
[7] J. Yang and J. Leskovec, “Overlapping community detection at scale: A nonnegative matrix factorization approach,” in Proc. 6th ACM Int. Conf. Web. Search and Data Mining (WSDM), Rome, Italy, Feb. 2013, pp. 587–596.
[8] M. E. Newman, “Spectral methods for community detection and graph partitioning,” Phys. Rev. E, vol. 88, no. 4, p. 042822, Oct. 2013.
[32] A. Clauset, C. R. Shalizi, and M. E. J. Newman, “Power-Law Distributions in Empirical Data,” SIAM Rev., vol. 51, no. 4, pp. 661–703, Feb. 2009.

[33] T. H. McCormick and M. J. Salganik, “How many people you know?: Efficiently estimating personal network size,” J. Am. Stat. Assoc., vol. 105, no. 489, pp. 59–70, Sep. 2010.

[34] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, “Kronecker graphs: An approach to modeling networks,” J. Mach. Learn. Research, vol. 11, pp. 985–1024, Feb. 2010.

[35] P. M. Weichsel, “The kronecker product of graphs,” J. Am. Math. Soc., vol. 13, no. 1, pp. 47–52, 1962.

[36] L. Katz, “A new status index derived from sociometric analysis,” Psychometrika, vol. 18, no. 1, pp. 39–43, Jul. 1953.

[37] C.-J. Hsieh and I. S. Dhillon, “Fast coordinate descent methods with variable selection for non-negative matrix factorization,” in Proc. 17th Int. Conf. Knowl. Disc. and Data Mining (SIGKDD), San Diego, CA, Aug. 2011, pp. 1064–1072.

[38] S. Thurner and S. Poledna, “DebtRank-transparency: Controlling systemic risk in financial networks,” Sci. Rep., vol. 3, no. 1888, May. 2013.

[39] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge University Press, 2004.

[40] V. Y. Tan and C. F. F ´evotte, “Automatic relevance determination in nonnegative matrix factorization,” in Proc. Workshop on Signal Process. Adapt. Sparse Structured Represent. (SPARS’09), Saint-Malo, France, 2009.

[41] E. Richard, P.-A. Savalle, and N. Vayatis, “Estimation of simultaneously sparse and low rank matrices,” in Proc. 29th Int. Conf. Machine Learning (ICML’12), Edinburgh, Scotland, Jun.-Jul. 2012, pp. 51–58.

[42] A. Lancichinetti and S. Fortunato, “Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities,” Phys. Rev. E, vol. 80, no. 1, p. 016118, Apr. 2009.

[43] A. Lancichinetti, S. Fortunato, and J. Kert ´esz, “Detecting the overlapping and hierarchical community structure in complex networks,” New J. Phys., vol. 11, no. 3, p. 033015, Mar. 2009.

[44] C. E. Shannon and W. Weaver, The mathematical Theory of Communication. University of Illinois Press, 1998.

[45] J. Leskovec and A. Krevl, “SNAP datasets: Stanford large network dataset collection,” Jun. 2014. [Online]. Available: http://snap.stanford.edu/data.

[46] J. Yang and J. Leskovec, “Defining and evaluating network communities based on ground-truth,” Knowl. and Inf. Syst., vol. 42, no. 1, pp. 181–213, 2015.

[47] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, “Group formation in large social networks: membership, growth, and evolution,” in Proc. 12th ACM Int. Conf. Knowl. Disc. and Data Mining (SIGKDD), Philadelphia, PA, Aug. 2006, pp. 44–54.

[48] C. E. Shannon and W. Weaver, The mathematical Theory of Communication. University of Illinois Press, 1998.