Predicting and Identifying Missing Node Information in Social Networks

Ron Eyal, Bar-Ilan University
Avi Rosenfeld, Jerusalem College of Technology
Sigal Sina, Bar-Ilan University
Sarit Kraus, Bar-Ilan University

0

In recent years, social networks have surged in popularity. One key aspect of social network research is identifying important missing information which is not explicitly represented in the network, or is not visible to all. To date, this line of research typically focused on finding the connections that are missing between nodes, a challenge typically termed as the *Link Prediction Problem*.

This paper introduces the **Missing Node Identification** problem where missing members in the social network structure must be identified. In this problem, indications of missing nodes are assumed to exist. Given these indications and a partial network, we must assess which indications originate from the same missing node and determine the full network structure.

Towards solving this problem, we present the **MISC** Algorithm (Missing node Identification by Spectral Clustering), an approach based on a spectral clustering algorithm, combined with nodes' pairwise affinity measures which were adopted from link prediction research. We evaluate the performance of our approach in different problem settings and scenarios, using real life data from Facebook. The results show that our approach has beneficial results and can be effective in solving the Missing Node Identification Problem. In addition, this paper also presents **R-MISC** which uses a sparse matrix representation, efficient algorithms for calculating the nodes' pairwise affinity and a proprietary dimension reduction technique, to enable scaling the **MISC** algorithm to large networks of more than 100,000 nodes. Last, we consider problem settings where some of the indications are unknown. Two algorithms are suggested for this problem - **Speculative MISC**, based on **MISC**, and **Missing Link Completion**, based on classical link prediction literature. We show that **Speculative MISC** outperforms **Missing Link Completion**.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications- Data mining

General Terms: Algorithms, Performance, Theory

Additional Key Words and Phrases: Social networks, spectral clustering, missing nodes

ACM Reference Format:
Ron Eyal, Avi Rosenfeld, Sigal Sina and Sarit Kraus, 2013. Predicting and Identifying Missing Node Information in Social Networks. *ACM Trans. Embedd. Comput. Syst.* 0, 0, Article 0 (2013), 26 pages.
DOI : http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Social Networks enable people to share information and interact with each other. These networks are typically formally represented as graphs where nodes represent people and edges represent some type of connection between these people [Liben-Nowell and Kleinberg 2007], such as friendship or common interests. These networks have become a key Internet application, with examples including popular websites such as Facebook, Twitter and LinkedIn.

Because of their ubiquity and importance, scientists in both academia and industry have focused on various aspects of social networks. One important factor that is often studied is the structure of these networks [Clauset et al. 2008; Eslami et al. 2011; Fortunato 2010; Freno et al. ; Gomez-Rodriguez et al. 2012; Gong et al. 2011; Kim and Leskovec 2011; Leroy et al. 2010; Liben-Nowell and Kleinberg 2007; Lin et al. 2012; Porter et al. 2009; Sadikov et al. 2011]. Previously, a *Link Prediction Problem* [Clauset et al. 2008; Liben-Nowell and Kleinberg 2007] was defined as attempting to locate which connections (edges) will soon exist between nodes. In this problem setting, the nodes of the network are known, and unknown links are derived from existing network information, including complete node information. In contrast, we consider a new *Missing Node Identification* problem which attempts to locate and iden-

This research is based on work supported in part by MAFAT. Sarit Kraus is also affiliated with UMIACS. Preliminary results were published in the AAAI 2011 paper entitled 'Identifying Missing Node Information in Social Networks'.
Author's addresses: R. Eyal, S. Sina and S. Kraus, Computer Science Department, Bar Ilan University, Ramat-Gan, Israel 92500
A. Rosenfeld, Department of Industrial Engineering, Jerusalem, Israel, 91160

DOI : http://dx.doi.org/10.1145/0000000.0000000

tify missing nodes within the network. This problem is significantly more difficult than the previously studied Link Prediction Problem as neither the nodes nor their edges are known with certainty.

To understand the importance of the missing node identification problem we introduce, please consider the following example. A hypothetical company, Social Games Inc., is running an online gaming service within Facebook. Many Facebook members are subscribers of this company's services, yet it would like to expand its customer base. As a service provider, Social Games maintains a network of users, which is a subset of the group of Facebook users, and the links between these users. The Facebook users who are not members of the service are not visible to their systems. Social Games Inc. would like to discover these Facebook nodes, and try to lure them into joining their service. The company thus faces the missing node identification problem. By solving this problem, Social Games Inc. could improve its advertising techniques and aim at the specific users who haven't yet subscribed to their service.

The above example exemplifies just one possible application of the missing node identification problem. In addition to commercial motivation, solving this problem could also be useful for personal interests and entertainment applications. This research direction has also particularly interested the security community. For example, missing nodes in some networks might represent missing persons that are sought after by family members who wish to know their full family tree or people wanted by the police as suspects in a crime. As a result, solving the missing node identification problem can be of considerable importance.

We focus on a specific variation of the missing node problem where the missing nodes requiring identification are "friends" of known nodes. An unrecognized friend is associated with a "placeholder" node to indicate the existence of this missing friend. Thus, a given missing node may be associated with several "placeholder" nodes, one for each friend of this missing node. We assume that tools such as image recognition software or automated text analysis can be used to aid in generating placeholder nodes. For example, a known user might have many pictures with the same unknown person, or another user might constantly blog about a family member who is currently not a member of the network. Image recognition or text mining tools can be employed on this and all nodes in the social network in order to obtain indications of the existence of a set of missing nodes. Placeholders can then be used to indicate where these missing nodes exist. However, it is likely that many of these placeholders are in fact the same person. Thus, our focus is on solving the identification of the missing nodes, and we frame the problem as: Given a set of placeholders, which of these placeholders do in fact represent the same person. Considering the assumption that placeholders are received from external data mining modules, this work is mainly focused on relatively small networks with a small number of missing nodes. Nonetheless, our proposed algorithm easily scales to networks of more than a hundred thousand nodes.

In this paper, we present a general method, entitled **MISC (Missing node Identification by Spectral Clustering)**, for solving this problem. This method relies on a spectral clustering algorithm previously considered only for other problems [Almog et al. 2008; Ng et al. 2001]. One key issue in applying the general spectral clustering algorithm is defining a measure for identifying similar nodes to be clustered together. Towards solving this issue, we present five measures for judging node similarity, also known as affinity. One of these measures is the Gaussian Distance Measure, typically used over Euclidean spaces in spectral clustering [Ng et al. 2001], while the other four measures are non-Euclidean measures which have been adapted from a related Link Prediction Problem [Liben-Nowell and Kleinberg 2007]. We found that the latter measures are especially useful in solving the missing node identification problem.

We begin this study in the next section by providing background of the spectral clustering algorithm and relevant social network research that relate to our solution for the Missing Node Identification problem. Then, in Section III, we formally define the Missing Node Identification problem. In Section IV, we detail how the MISC algorithm can be applied, including how different node affinity measures inspired by the Link Prediction Problem can be applied within our solution. Section V presents our evaluation dataset and methodology, and Section VI provides an empirical analysis of MISC's performance when the different affinity measures are used. Following this analysis, we provide two extensions to our base algorithm. In Section VII we describe how our base algorithm can be extended to handle large networks of hundreds of thousands of nodes. In Section VIII we describe the comparison between the MISC algorithm and KronEM [Kim and Leskovec 2011], a state-of- the-art algorithm for the prediction of missing nodes. In Section IX we suggest methods for modifying our base approach to address situations when partial information about the number of missing nodes is unknown, or not all of the placeholders are known. Under the problem setting of unknown placeholders, we are able to show that the MISC algorithm, based on spectral clustering, outperforms 'Missing Link Completion', an algorithm based on classical solutions to the Link Prediction Problem. Section X concludes and provides directions for future research.

2. RELATED WORK

In solving the Missing Node Identification problem, this research aims to use variations of two existing research areas – spectral clustering algorithms and metrics built for the Link Prediction Problem. The spectral clustering algorithm of Jordan, Ng and Weiss [Ng et al. 2001] is a well documented and accepted algorithm, with applications in many fields

including statistics, computer science, biology, social sciences and psychology [von Luxburg 2007]. While spectral clustering algorithms have been applied to many areas, its use in the Missing Node Identification problem has not been previously considered and cannot be directly applied from previous works.

The main idea behind the spectral clustering framework is to embed a set of data points which should be clustered in a graph structure in which the weights of the edges represents the *affinity* between each pair of points. Two points which are "similar" will have a larger value for the weight of the edge between these two points' than two points which are "dissimilar". The similarity, or affinity, function is used to calculate an affinity matrix which describes the pairwise similarity between points. The *affinity matrix* can be thought of as a weighted adjacency matrix, and therefore it is equivalent to an *affinity graph*. After generating the affinity graph, an approximation of the min-cut algorithm is employed in order to split the graph into partitions which maximize the similarity between intra-partition points and minimize the similarity between points in different partitions. This is done by manipulating the affinity matrix representing the affinity graph and solving an eigenvector problem.

While the general framework of this algorithm is to embed points in a Euclidean space as an affinity graph, our approach is to construct the affinity graph directly from the social network graph structure. The key challenge in applying the spectral clustering algorithm to the Missing Node Identification problem is how to compute the level of similarity between nodes in the social network while constructing the affinity matrix. Towards defining this measure, we consider adopting measures developed for a related problem, the Link Prediction Problem. In the Link Prediction Problem there are a set of known nodes, and the goal is to discover which connections, or edges, will be made between nodes [Clauset et al. 2008; Liben-Nowell and Kleinberg 2007]. In contrast, in the missing node problem, even the nodes themselves are not known, making the problem significantly more difficult. Nonetheless, we propose a solution where measures originally used to solve the Link Prediction Problem are utilized to form the affinity matrix in the first step of solving this problem as well.

Various methods have been proposed to solve the Link Prediction Problem. Approaches typically attempt to derive which edges are missing by using measures to predict link similarity based on the overall structure of the network. However, these approaches differ as to which computation is best suited for predicting link similarity. For example, Liben-Nowell and Kleinberg [Liben-Nowell and Kleinberg 2007] demonstrated that measures such as the shortest path between nodes and different measures relying on the number of common neighbors can be useful. They also considered variations of these measures, such as using an adaptation of Adamic and Adar's measure of the similarity between webpages [Adamic and Adar 2003] and Katz's calculation for shortest path information, which weight short paths more heavily [Katz 1953] than the simpler shortest path information. After formally describing the missing node identification problem, we detail how the spectral clustering algorithm can be combined with these link prediction methods in order to effectively solve the missing node identification problem.

Many other studies have researched problems of missing information in social networks. Guimera and Sales-Pardo [Guimerà and Sales-Pardo 2009] propose a method which performs well for detecting missing links as well as spurious links in complex networks. The method is based on a stochastic block model, where the nodes of the network are partitioned into different blocks, and the probability of two nodes being connected depends only on the blocks to which they belong. Some studies focus on understanding the propagation of different phenomena through social networks as a diffusion process. These phenomena include viruses and infectious diseases, information, opinions and ideas, trends, advertisements, news and more. Gomez-Rodriguez et al. [Gomez-Rodriguez et al. 2012] attempted to infer a network structure from observations of a diffusion process. Specifically, they observed the times when nodes get infected by a specific contagion, and attempted to reconstruct the network over which the contagion propagates. The reconstruction is done through the edges of the network, while the nodes are known in advance. Eslami et al. [Eslami et al. 2011] studied the same problem. They modeled the diffusion process as a Markov random walk and proposed an algorithm called DNE to discover the most probable diffusion links.

Sadikov et al. [Sadikov et al. 2011] also studied the problem of diffusion of data in a partially observed social network. In their study they proposed a method for estimating the properties of an information cascade, the nodes and edges over which a contagion spreads through the network, when only part of the cascade is observed. While this study takes into account missing nodes and edges from the cascade, the proposed method estimates accumulative properties of the true cascade and does not produce a prediction of the cascade itself. These properties include the number of nodes, number of edges, number of isolated nodes, number of weakly connected components and average node degree.

Other works attempted to infer missing link information from the structure of the network or information about known nodes within the network. For example, Lin et al. [Lin et al. 2012] proposed a method for community detection, based on graph clustering, in networks with incomplete information. In these networks, the links within a few local regions are known, but links from the entire network are missing. The graph clustering is performed using an iterative algorithm named DSHRINK. Gong et al. [Gong et al. 2011] proposed a model to jointly infer missing links and missing node attributes by representing the social network as an augmented graph where attributes are also represented by

nodes. They showed that link prediction accuracy can be improved when first inferring missing node attributes. Freno et al. [Freno et al.] proposed a supervised learning method which uses both the graph structure and node attributes to recommend missing links. A preference score which measures the affinity between pairs of nodes is defined based on the feature vectors of each pair of nodes. Their algorithm learns the similarity function over feature vectors of the graph structure. Kossinets [Kossinets 2003] assessed the effect of missing data on various networks and suggested that nodes may be missing, in addition to missing links. In this work, the effects of missing data on network level statistics were measured and it was empirically shown that missing data causes errors in estimating these parameters. While advocating its importance, this work does not offer a definitive statistical treatment to overcome the problem of missing data.

One can divide studies of the missing links problem into two groups: unsupervised methods and supervised ones. Unsupervised methods include works such as those done by Liben-Nowell and Kleinberg [Liben-Nowell and Kleinberg 2007], Katz [Katz 1953], and Gong et al. [Gong et al. 2011]) which do not require the inputted data to be labeled in any way. Instead, these methods learn through implying information based on the specific structure of the input network. A second group of methods require some element of tagged data be inputted in a training phase where the data is typically explicitly labeled so that the classifier can be created. These methods include the works of Freno et al. [Freno et al.] and Gong et al. [Gong et al. 2011]) who use a binary classifier in order to predict the missing links. Although the supervised methods often yields better results, it requires an additional step of manually tagging the inputted data– something that requires additional resources and time. Additionally, this approach assumes some similarity between the training dataset and test dataset. In our work, we intentionally avoided basing ourselves on supervised methods as we wished to present an method free of these steps and assumptions and thus opted to create an unsupervised method.

Most similar to our paper is the recent work by Kim and Leskovec [Kim and Leskovec 2011], which also tackled the issue of missing nodes in a network. This work deals with situations where only part of the network is observed, and the unobserved part must be inferred. The proposed algorithm, called KronEM, uses an Expectation Maximization approach, where the observed part is used to fit a Kronecker graph model of the network structure. The model is used to estimate the missing part of the network, and the model parameters are then reestimated using the updated network. This process is repeated in an iterative manner until convergence is reached. The result is a graph which serves as a prediction of the full network. This research differs from ours in several aspects. First, while the KronEM prediction is based on link probabilities provided by the EM framework, our algorithm is based on a clustering method and graph partitioning. Secondly, our approach is based on the existence of missing node indications obtained from data mining modules such as image recognition. When these indications exist, our algorithm can be directly used to predict the original graph. As a result, while KronEM is well suited for large scale networks with many missing nodes, our algorithm may be effective in local regions of the network, with a small number of missing nodes, where the data mining can be employed. The results of experiments performed in this study show that our proposed algorithm, MISC, can achieve better prediction quality than KronEM, thanks to the use of indications of missing nodes.

3. FORMAL DEFINITION OF THE MISSING NODE IDENTIFICATION PROBLEM

We formally define the new Missing Node Identification Problem as follows. Assume that there is a social network $G = (V, E)$ in which $e = \langle v, u \rangle \in E$ represents an interaction between $v \in V$ and $u \in V$. Some of the nodes in the network are missing and are not known to the system. We denote the set of missing nodes $V_m \subset V$, and assume that the number of missing nodes is given as $N = |V_m|$. We denote the rest of the nodes as known, i.e., $V_k = V \setminus V_m$, and the set of known edges is $E_k = \{\langle v, u \rangle \mid v, u \in V_k \wedge \langle v, u \rangle \in E\}$, i.e. only the edges between known nodes.

Towards identifying the missing nodes, we focus on a part of the network, $G_a = \langle V_a, E_a \rangle$, that we define as being available for the identification of missing nodes. In this network, each of the missing nodes is replaced by a set of placeholders. Formally, we define a set V_p of placeholders and a set E_p for the associated edges. For each missing node $v \in V_m$ and for each edge $\langle v, u \rangle \in E$, $u \in V_k$, a placeholder is created. That is, we add a placeholder for v as v' to V_p and for the original edge $\langle v, u \rangle$ we add a placeholder $\langle v', u \rangle$ to E_p. We denote the origin of $v' \in V_p$ with $o(v')$. Putting all of these components together, $V_a = V_k \cup V_p$ and $E_a = E_k \cup E_p$. In this setting, both the known nodes and the placeholders, along with their associated edges are defined as being the portion of the network, G_a, that is available for the missing nodes identification process. The problem is that for a given missing node v there may be many placeholders in V_p. The challenge is to try to determine which of the placeholders are associated with v. This will allow us to reconstruct the original social network G.

Formally, we define the missing node identification problem as: Given a known network $G_k = \langle V_k, E_k \rangle$, an available network $G_a = \langle V_a, E_a \rangle$ and the number of missing nodes N, divide the nodes of $V_a \setminus V_k$ to N disjoint sets $V_{a_1}, \ldots, V_{a_N}$ such that $V_{a_i} \subseteq V_p$ are all the placeholders of $v_i \in V_m$.

To better understand this formalization, consider the following example. Assume Alice is a Facebook member, and thus is one of the available nodes in V_a. She has many known social nodes (people), V_k, within the network,

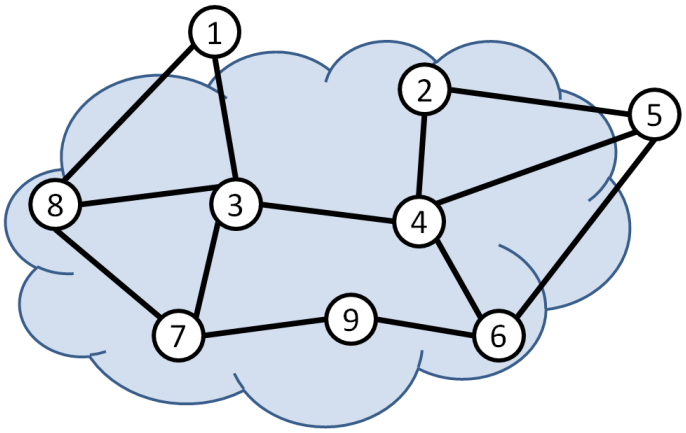

Fig. 1: Illustration of the missing node identification problem. The whole graph represents the original network, $G = \langle V, E \rangle$. Nodes 1 and 5 are missing nodes. The sub-graph containing the nodes which appear inside the cloud is the known network, $G_k = \langle V_k, E_k \rangle$.

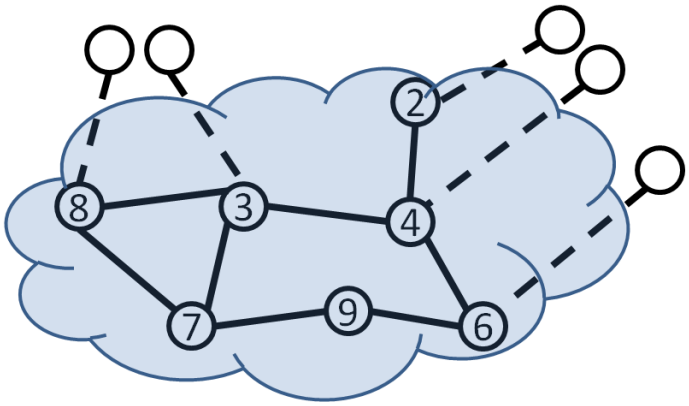

Fig. 2: Illustration of the available network obtained by adding placeholders for the missing nodes. The nodes which appear outside of the cloud are the placeholders. The sub-graph containing the nodes which appear inside the cloud is still the known network, $G_k = \langle V_k, E_k \rangle$. The whole graph represents the available network, $G_a = \langle V_a, E_a \rangle$, which includes the known network and the placeholders.

but her cousin Bob is not a member of Facebook. Bob is represented by a missing node $w \in V_m$. From analysis of text in her profile we might find phrases like "my long lost cousin Bob", indicating the existence of the missing node representing Bob. Alternately, from examining profile pictures of Alice's friends, an image recognition software package identifies pictures of an unknown male person, resulting in another indication of the existence of a missing node. Each indication is represented by a placeholder node $v' \in V_p$ and a link $(u, v') \in E_p$, where u is the known node (e.g., Alice) which contains the indication. By solving the missing node identification problem we aim to identify which of these indications point to the same missing node, representing Bob, and which represent other missing nodes.

4. THE MISC ALGORITHM

The main challenge in solving the missing node problem is discovering how to correctly identify the set of V_p placeholders as the set of V_m missing nodes. As we are limited to general structural knowledge about the portion of the network G_a that is available, solving this problem is far from trivial. Our key approach is to use a clustering approach, and specifically spectral clustering, to group V_p into N clusters in order to identify the missing nodes. The challenge to using any clustering algorithm, including spectral clustering, is how to provide the clustering algorithm with a defined Euclidean value that quantifies distances between both the known portion of the network and the placeholders representing the missing nodes. To better understand this difficulty, this section first briefly introduces spectral clustering and focuses on the challenges in applying it to the missing node identification problem. Our novel contribution is that we apply a series of affinity measures primarily based on literature from the related link prediction problem which can

be used to solve the missing node problem. These affinity measures are critical in providing a non-discrete measure between both known nodes and placeholders such that the clustering algorithm can operate on that input. Finally, we present our novel MISC algorithm for solving the missing node identification problem based on combining both the spectral clustering algorithm and these affinity measures.

4.1. Spectral Clustering

Spectral clustering is a general algorithm used to cluster data samples using a certain predefined similarity or affinity measure between them. The algorithm creates clusters which maximize the similarity between points in each cluster and minimize the similarity between points in different clusters. This algorithm accepts as its input a set of M sample coordinates in a multi-dimensional space, described by a matrix S. The algorithm also requires the number of clusters, K, be known as used as input. In its original form, the algorithm constructs an affinity matrix which describes the affinity between each pair of samples based on the Euclidean distance on the multi-dimensional space. This affinity matrix is equivalent to an affinity graph, where each data sample is represented by a node and the affinity between two samples is represented by the weight of the edge between the two matching nodes. Spectral clustering solves the minimum cut problem by partitioning the affinity graph in a manner that maximizes the affinity within each partition and minimizes the affinity between partitions. Since finding this partitioning is NP-Hard, spectral clustering finds an approximated solution to this problem. While the reader is encouraged to review the algorithm in its entirety [Ng et al. 2001], a brief description of this algorithm is presented below in Algorithm 1.

ALGORITHM 1: - Spectral Clustering

Input: S - data samples in a multi-dimensional space

K - number of clusters

σ - standard deviation for the Gaussian distance function

Output: C $\in \mathbb{N}^{1\times|S|}$ - a vector indicating a cluster index for each sample in S

1: Define s_i to be the coordinates of every sample i in the multi-dimensional space and calculate an affinity matrix $\mathbf{A} \in \mathbb{R}^{M\times M}$, where M is the number of samples. **A** defines the affinity between all pairs of samples (i, j) using the Gaussian distance function:
$A_{ij} = exp(-||s_i - s_j||^2/2\sigma^2)$

2: Define D to be the diagonal matrix whose (i, i) element is the sum of A's i-th row, and construct the matrix L:
$L = D^{-1/2}AD^{-1/2}$

3: Find x_1,x_2,...,x_K, the K largest eigenvectors of L (chosen to be orthogonal to each other in the case of repeated eigenvalues), and form the matrix X = $[x_1x_2...x_K] \in \mathbb{R}^{M\times K}$ by stacking the eigenvectors in columns.

4: Form the matrix Y from X by renormalizing each of X's rows to have unit length.

5: Treating each row of Y as a point in $\mathbb{R}^K$, cluster them into K clusters via k-means or any other clustering algorithm.

6: Assign the original sample i to cluster j if and only if row i of the matrix Y was assigned to cluster j.

The key to the success of Algorithm 1 is constructing the affinity matrix in the first step of the algorithm. This matrix must represent the pairwise affinity as accurately as possible according to the given problem setting. Note that this algorithm assumes data nodes residing in some Euclidean space and thus defines affinity between samples accordingly. However, in the case of social network graphs, it is very difficult to embed the nodes of the graph in a Euclidean space as defined by the original algorithm. This is because it is unclear if such a space can be defined which represents the actual distance between nodes in the graph.

To understand this difficulty, consider the triangle inequality which is one of the basic characteristics of a Euclidean space. This inequality may not hold for many reasonable distance metrics between nodes. For instance, consider a distance metric that incorporates the number of common neighbors between two nodes. While nodes a and b may not have any common neighbors, causing their distance to be infinite, they may both have common neighbors with node c. The common neighbors measure for this example does not obey the triangle inequality because under this measure d(a,b) > d(a,c) + d(c,b). In general, defining a space which obeys the triangle inequality is difficult because it requires an examination of more than two nodes at once when defining their distances. A distance metric which does obey this inequality is the shortest path length between two nodes, which we define later. Nonetheless, as we have found, this measure does not necessarily yield the best results for missing node identification. In order to bypass the Euclidean space difficulty, we have decided to alter the first step of the algorithm and to define direct, graph-based measures for building the affinity matrix A, rather than using the Euclidean space. While applying this algorithm to missing node identification, we need to address how this measure can be calculated to represent affinity between nodes in a social network. Due to the complexity of social networks, and the need to compute this measure in a multi-dimensional space, calculating this measure is far from trivial. We consider several different methods, most of which have been

proven to be useful for solving the Link Prediction Problem in social networks [Adamic and Adar 2003; Katz 1953; Liben-Nowell and Kleinberg 2007]. We empirically compare these methods with the Euclidean method described in the original algorithm. These methods are discussed in detail in the next section.

4.2. Incorporating Link Prediction Measures

The key novelty within this paper is that we propose that affinity measures be constructed based on general graph measures or previously developed measures for the related Link Prediction Problem [Adamic and Adar 2003; Katz 1953; Liben-Nowell and Kleinberg 2007]. Specifically, we discuss how five such measures can be potentially applied to the spectral clustering algorithm [Ng et al. 2001]:

(1) **Gaussian Distance:** Define D_{ij} to be the length of the shortest path between nodes i and j. Define D_i to be the vector of the length of shortest paths from node i to all other nodes. Calculate A_{ij} as in step 1 of the original spectral clustering algorithm:
$A_{ij} = exp(-||D_i - D_j||^2/2\sigma^2)$.
(2) **Inverse Squared Shortest Path (ISSP)**:
$A_{ij} = \frac{1}{(D_{ij})^2}$ where D_{ij} is the length of the shortest path between nodes i and j.
(3) **Relative Common Friends:**
$A_{ij} = \frac{|\Gamma(i) \bigcap \Gamma(j)|}{min(|\Gamma(i)|,|\Gamma(j)|)}$ where $\Gamma(i)$ is defined as the group of neighbors of node i in the network graph.
(4) **Adamic / Adar:**
$A_{ij} = \sum_{k \in \Gamma(i) \bigcap \Gamma(j)} \frac{1}{log(|\Gamma(k)|)}$
(5) **Katz Beta:**
$A_{ij} = \sum_{k=1}^{\infty} \beta^k \cdot$ (number of paths between nodes i and j of length exactly k).

All five of these measures present possible similarity values for creating the affinity matrix, A_{ij}, in the spectral clustering algorithm. The Gaussian Distance is based on the standard distance measure often used in spectral clustering [von Luxburg 2007]. The Inverse Squared Shortest Path (ISSP) measure is based on the length of the shortest path between two points, i and j, here representing two nodes in the social network. This measure presents an alternative to the original Gaussian Distance used to measure Euclidean distances, and is not directly inspired by the Link Prediction Problem. In contrast, the next three measures were directly inspired by this literature. The Relative Common Neighbors measure checks the number of neighbor nodes that i and j have in common ($|\Gamma(i) \bigcap \Gamma(j)|$). We divide by $min(|\Gamma(i)|, |\Gamma(j)|)$ in order to avoid biases towards nodes with a very large number of neighbors. Similarly, the Adamic / Adar measure also incorporates the common neighbor measure: ($\Gamma(i) \bigcap \Gamma(j)$), checking the overall connectivity of each common neighbor to other nodes in the graph and giving more weight to common neighbors who are less connected. Since the nodes that act as placeholders for missing nodes only have one neighbor each, the common neighbors and the Adamic/Adar measures do not represent these nodes well. Therefore, for these measures only, we also consider them to be connected to their neighbor's neighbors. Last, the Katz measure [Katz 1953] directly sums the number of paths between i and j, using a parameter β which is exponentially damped by the length of each path. Similar to the common neighbor measure, this measure also stresses shorter paths more heavily. Finally, for evaluating these algorithms, we consider a baseline Random Assignment algorithm that assigns each placeholder to a random cluster and represents the most naive of assignment algorithms.

4.3. Solving the Missing Node Identification Problem

Generally, our solution for solving the missing node problem is based on the following three steps:

(1) Cluster the placeholder nodes, using a spectral clustering approach, thus creating disjointed groups of placeholders which have a high probability of representing the same missing node.
(2) Unite all of the placeholders from each cluster into one node, attempting to predict the missing node that is most likely represented by the placeholders in that cluster.
(3) Connect the predicted node to all neighbors of the placeholders that were in the matching cluster, in the hopes of creating a graph that is as similar as possible to the original graph G.

Clustering the placeholders is the crucial part of this approach. When this clustering is performed perfectly, the placeholders in each cluster will all originate from the same missing node. As a result, the predicted node created from each cluster will match a specific missing node from the original network graph from which the missing nodes were removed. In this case the predicted graph will be identical to the original graph. When clustering is not fully successful, some clusters will contain placeholders which originate from different missing nodes, the predicted nodes will not exactly match the missing nodes and the predicted graph will be less similar to the original graph.

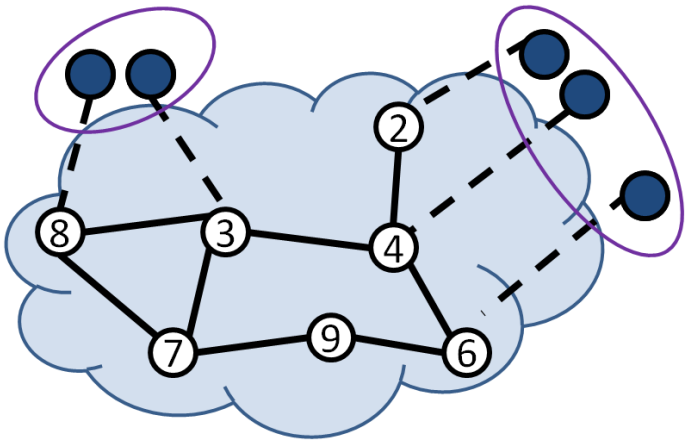

Fig. 3: Correct clustering of the placeholders for missing nodes 1 and 5. The placeholders in each cluster are united to one node which represents a missing node.

Specifically, Algorithm 2, MISC, presents how this solution is accomplished. This algorithm accepts the known and available parts of the network graph, as described in the problem definition. We also assume at this point that the number of missing nodes, N, is given. N is used to define the number of clusters that the spectral clustering algorithm will create. The final input is α, a procedure for calculating the pairwise affinity of nodes in the graph. An example of such a procedure could be one that implements the calculation of one of the affinity measures adopted from the Link Prediction Problem.

ALGORITHM 2: - MISC (Missing node Identification by Spectral Clustering)

Input: $G_k = \langle V_k, E_k \rangle$ - the known part of the network
$G_a = \langle V_a, E_a \rangle$ - the available part of the network
N – the number of missing nodes
$\alpha : G(V, E) \longrightarrow \mathbb{R}^{|V| \times |V|}$ - a procedure for calculating the affinity matrix of nodes in a graph

Output: $C \in \mathbb{N}^{|V_a \setminus V_k|}$ - a vector indicating the cluster index of each placeholder node, $\hat{G} = \left(\hat{V}, \hat{E}\right)$ – prediction of the full network graph

---

1: $\mathbf{A} \in \mathbb{R}^{|V_a| \times |V_a|} \longleftarrow \alpha(G_a)$ - calculate the affinity matrix of the available nodes in the graph
2: Perform steps 2-4 of **Spectral Clustering Algorithm** (Algorithm 1) to calculate Y, using N as the input K to algorithm 1
3: Y' $\longleftarrow \{Y_i \mid v_i \notin V_k\}$ – keep only the rows of Y which match the placeholder nodes
4: C$\longleftarrow$ *k_means(Y',N)* – cluster the rows which match the placeholder nodes to N clusters
5: $\hat{V} \longleftarrow V_k$, $\hat{E} \longleftarrow E_k$ - initialize the output graph to contain the known network
6: For each cluster $c \in C$ create a new node $v_c \in \hat{V}$
7: For each placeholder v in cluster c and edge $(u, v) \in E_a$, create an edge $(u, v_c) \in \hat{E}$
8: Return $C, \hat{G} = \left(\hat{V}, \hat{E}\right)$

The first two steps of this algorithm are based on the spectral clustering algorithm and measures described in the previous two sections. This algorithm first calculates the affinity matrix of the available nodes using the given procedure α (step 1). Note that we use this algorithm with any one of the five affinity measures as described in the previous section. Next, steps 2-4 of the original spectral clustering algorithm are followed in order to calculate the matrix Y (step 2). This is a transformation that spectral clustering performs in order to transform the data points into a vector space in which k-means clustering can be employed. The number of missing nodes, N, is an input parameter in MISC. It is used as the number of clusters for the spectral clustering algorithm, which is marked as K. Even though we only need to cluster the placeholders, the matrix Y in the spectral clustering algorithm contains all of the M data samples as coordinates in an N dimensional space. Spectral clustering performs k-means clustering on the samples in this space. In the context of MISC, each row of Y corresponds to a node in G_a embedded in an N dimensional space.

Step 3 of the algorithm is specific to our problem. Here, the rows of Y which correspond to the known nodes in V_k are removed. As opposed to general clustering problems where all the data must be clustered in an unsupervised

manner, in our case most of the nodes are known and the challenge is to find a correct clustering for the placeholders. Therefore, for the sake of clustering and unifying only the placeholders, only the rows corresponding to placeholders in Y are kept.

On the other hand, the affinity between the known nodes and the placeholders contains important information which should be utilized. For this reason, all the nodes are embedded in the affinity matrix, and all the columns in Y remain. Notice that in this manner, the information obtained from the known nodes in the embedding process is still present in the matrix Y' in the form of the coordinates matching the placeholders (see step 3 of Algorithm 2).

In step 4 of Algorithm 2, the remaining rows, which correspond to placeholders, are clustered using k-means clustering. In this step also, N is used as the number of clusters K. This is because every cluster is used to predict a missing node in the following steps. According to spectral clustering theory [Ng et al. 2001], k-means clustering is expected to achieve better results when employed using the coordinates depicted in Y or Y' than when applied to the original data.

In steps 5-8 the predicted graph is created by uniting all of the placeholders in each cluster to a new node and connecting this node to the known nodes which were neighbors of the original placeholders in the corresponding cluster. Recall that each cluster represents a predicted missing node. The placeholders in that cluster each represent a link of the predicted missing node, and for this reason the links are created between the new node and the neighbors of the placeholders in the corresponding cluster. Another advantage of removing the known nodes in step 3 is clear at this point - each resulting cluster contains only placeholders, and can therefore represent a missing node from the original graph.

5. DATASET DESCRIPTION AND EVALUATION METHODOLOGY

To empirically study the Missing Node Identification problem and assess the proposed solutions, we must be able to simulate the problem setting using real world data. Within this section we first describe the dataset utilized and methods of synthesizing the problem. We then discuss the evaluations, methods and methodology used to compare the proposed solutions empirically.

5.1. Dataset Description

For an empirical evaluation of our method we use a previously developed social network dataset - the Facebook MHRW dataset [Gjoka et al. 2010]. This dataset contains structural information sampled from Facebook, including over 900,000 nodes and the links between them. For each node certain social characteristics are stored, such as shared academic networks (e.g. all people from Harvard University), corporate networks (e.g. workers in AIG), geographical networks (e.g. members from Idaho), or networks of people who share similar interests (e.g. love of chocolate). All nodes are anonymized as numbers without any indication of their true identity.

The main challenge we had to address in using a dataset of this size was processing the data within a tractable period and overcoming memory constraints. In order to create more tractable datasets, we considered two methods of creating subsets of the Facebook data [Gjoka et al. 2010]. In the first method, we create a subset based on naturally occurring similarities between nodes according to users' network membership characteristics within the social network. Each subset is created by sampling all the nodes in a specific user network and the links between these nodes. Nodes with only one link or no links at all are removed. The advantage of this method of creating the subsets is that there is a higher chance of affiliation between the nodes in the user network as compared to random nodes selected from the entire social network. However, the disadvantage is that the nodes which make up the user network may not be completely connected, for example if the user network is comprised of several disconnected components. In fact, the subgraph of nodes that is part of a specific user network may be very sparse. Another disadvantage is that the user networks in the dataset are limited in size and therefore large subgraphs cannot be created from them. In contrast, for the second method of creating a subset, we begin with the entire dataset, and extract a subset based on a BFS walk starting from a random node in the dataset. Here no previous information about the social network is necessary, but the BFS generated subset may not accurately represent the actual topology of the entire network.

In order to synthesize the missing node problem within these two subsets, we randomly mark N nodes as the set of missing nodes, V_m. We then remove these nodes from the network, and replace each link (v, u) between $v \in V_m$ and $u \in V_k$ with a placeholder node $v' \in V_p$ and a link $(v', u) \in E_p$. The resulting network G_a is the available network used as input to our proposed MISC algorithm for solving the Missing Node Identification problem.

5.2. Evaluation Measures

We considered two types of evaluation measures to measure the effectiveness of the solutions presented. One measure is a straightforward similarity measure that compares the output graph of the MISC algorithm, $\hat{G} = (\hat{V}, \hat{E})$, to the original network graph, G, from which the missing nodes were removed.

Within the first measure, we quantify the similarity of two graphs based on the accepted *Graph Edit Distance* (GED) measure [Bunke and Messmer 1993; Kostakis et al. 2011]. The GED is defined as the minimal number of edit operations required to transform one graph into the other. An edit operation is an addition or deletion of a node or an edge. Since finding the optimal edit distance is NP-Hard, we use a previously developed simulated annealing method [Kostakis et al. 2011] to find an approximation of the Graph Edit Distance. The main advantage of this method of evaluation is that it is independent of the method used to predict $\hat{G}$, making it very robust. For instance, GED can be used to compare two methods which might create a different number of clusters from each other, if the number of clusters is unknown in advance. GED can also be used to compare methods that cluster a different amount of placeholders, for example, if not all of the placeholders are given. In fact, it can be used to compare any two methods, as long as they both produce a predicted graph.

A disadvantage to computing the GED lies within its computational cost. This expensive cost exists because the GED measure is based on finding an optimal matching of the nodes in $\hat{G}$ to the nodes in G which minimizes the edit distance. Even if we take advantage of the fact that most of the nodes are known nodes from V_k and their matching is known, there are still N unknown nodes to be aligned - the missing nodes in G to the nodes generated from each cluster in $\hat{G}$. This allows for $N!$ possibilities. Therefore, a heuristic search method must be used, as we do, which may lead to over-estimations of the edit distance.

Due to this expense, a *purity* measure, which can be computed simply, can be used instead. The purity measure attempts to assess the quality of the placeholders' clustering. This can be done since we know in advance the correct clustering of the placeholders. Under the correct clustering, each cluster would group together the placeholders originating from the same missing node. A high quality algorithm would produce a clustering which is most similar to the true clustering of the placeholders. The clustering quality is tested using the purity measure which is often used to evaluate clustering algorithms [Strehl and Ghosh 2003]. This measure is calculated in the following manner:

(1) Classify each cluster according to the true classification of the majority of samples in that cluster. In our case, we classify each cluster according to the most frequent true original node $v \in V_m$ of the placeholder nodes in that cluster.
(2) Count the number of correctly classified samples in all clusters and divide by the number of samples. In our case the number of samples (nodes) that are classified is $|V_p|$.

Formally, in this problem setting, purity is defined as: $purity(C) = \frac{1}{|V_p|}\sum_k max_{v_j \in V_m} |c_k \cap \{v' \in V_p \mid o(v') = v_j\}|$ where c_k is defined as the set of placeholders which were assigned to cluster k. Note that as the number of missing nodes increases, correct clustering becomes more difficult, as there are more possible original nodes for each placeholder. As our initial results show, the purity indeed decreases as the number of missing nodes increases (see Figures 5 and 6). This evaluation method works well because it is easy to calculate and it reflects the quality of the algorithm well, when comparing clustering algorithms which create the same number of clusters. Its disadvantage is due to the fact that it can be biased depending on the number of clusters. If the same data is divided into a large number of clusters it is easier to achieve higher purity. Therefore this method is used only when the number of clusters is known in advance, and there is no advantage to selecting a large number of clusters.

5.3. Evaluation Methodology

In this subsection we describe our evaluation methodology and the experimental setup (see flowchart in Figure 4). The flow begins with the full Facebook MHRW dataset, containing over 900,000 nodes. This dataset is sampled by running BFS from a random node or by selecting a specific user network, as described in subsection V.A. When running BFS, the sampling is repeated ten times, each starting from a different random node. In other words, ten different networks are created for each size, in order to avoid randomization errors and biases. In the following step, N nodes are randomly selected as missing nodes. This is repeated several times from each one of the networks, depending on the experiment, resulting in different random instances of the problem setting for each combination of graph size and N. From each instance the randomly selected missing nodes are removed. Each missing node is replaced with a placeholder for each of its links to remaining nodes in the network. The resulting graph is fed into the MISC algorithm, which produces a clustering of the placeholders. By uniting the placeholders in each cluster into a new node and connecting the new node to the neighbors of the placeholders in the corresponding cluster, MISC creates a predicted network. This predicted network is supposed to resemble as closely as possible the structure of the original network from which the random nodes were removed. The clustering produced by MISC is evaluated using the purity measure described above, and the average purity value achieved for all of the instances is reported. The Graph Edit Distance is also evaluated by comparing the predicted network to the original network. This result is also averaged over the different instances and is reported as well. In each experiment report we indicate the number of iterations performed. Nearly all points in the graphs of the relevant figures below are the average of at least 100 runs

of the algorithm on randomly generated configurations. However, for highly time consuming experiments, such as the comparison with the KronEM algorithm, only 40 iterations of the experiment were run. Nevertheless, the results were significant even with this number of iterations.

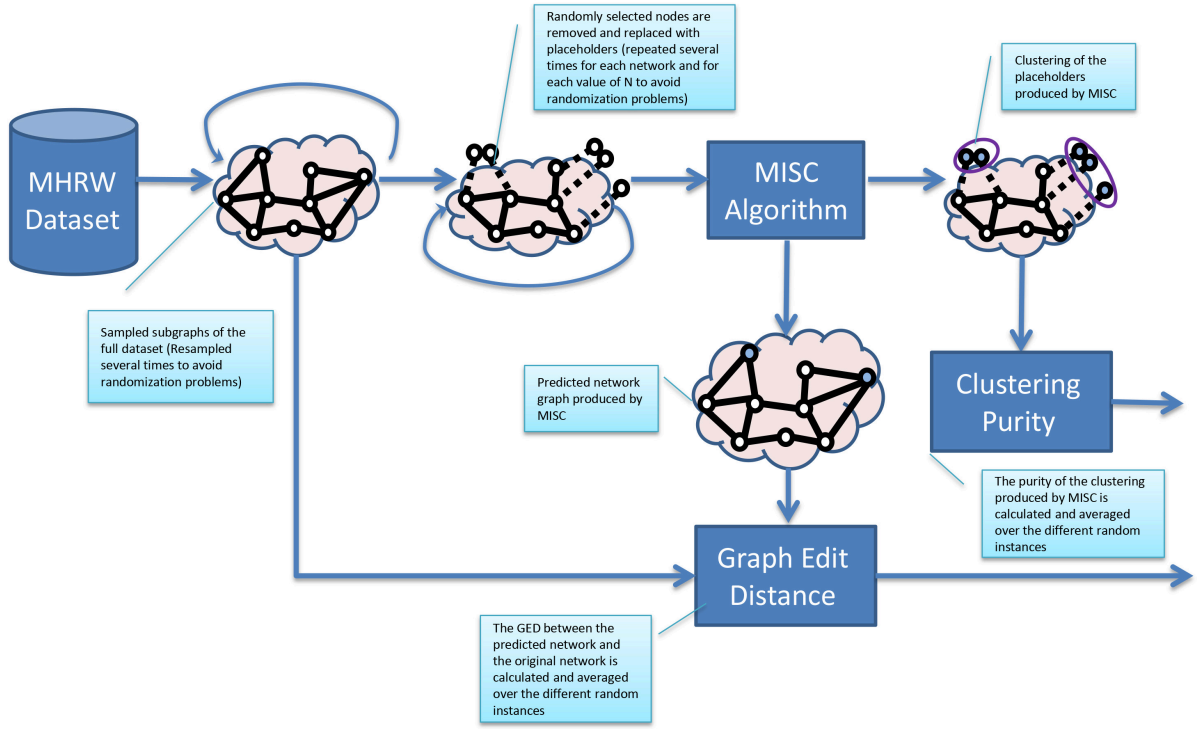

Fig. 4: A figure explaining the evaluation methodology for the experiments within this work.

6. COMPARING AFFINITY MEASURES

To assess the quality of our algorithm we ran experiments on the subgraphs obtained from the Facebook dataset. Each experiment consisted of synthesizing the missing node problem as described in Section 5, and running the spectral clustering algorithm using each one of the proposed affinity measures. We measured the purity achieved when using each of the affinity measures and compared the results to the purity of a random clustering, where each placeholder was randomly assigned to a cluster. Each experiment described in this section was repeated over 150 times, each time with randomly selected missing nodes. The results displayed indicate the average over all experiments.

The results obtained from these experiments clearly show that the MISC algorithm obtained significantly better results than the random clustering, regardless of which affinity measure was used, showing the success of this method in solving the missing node identification problem. We also compared the relative success of the different affinity measures described in Section IV.B. We find that Inverse Squared Shortest Path, Relative Common Neighbors and Adamic / Adar performed slightly better than Gaussian Distance and Katz Beta measures in this dataset. Figure 5 shows the purity achieved when using each of the affinity measures with the spectral clustering algorithm, on fifteen subgraphs containing 2,000 nodes each, obtained by BFS from a random node in the full dataset. Each point in the figure represents the average purity from the experiments described above. MISC performed much better than the random clustering with each one of the affinity measures. When comparing the affinity measures, it seems that the Inverse Squared Shortest Path was slightly better than the others. The same experiments were performed on subgraphs of user networks taken from the full facebook dataset. The following networks were selected:

(1) Network 362 – containing 3,189 nodes and 3,325 undirected links
(2) Network 477 – containing 2,791 nodes and 3,589 undirected links
(3) Network 491 – containing 2,874 nodes and 3,143 undirected links

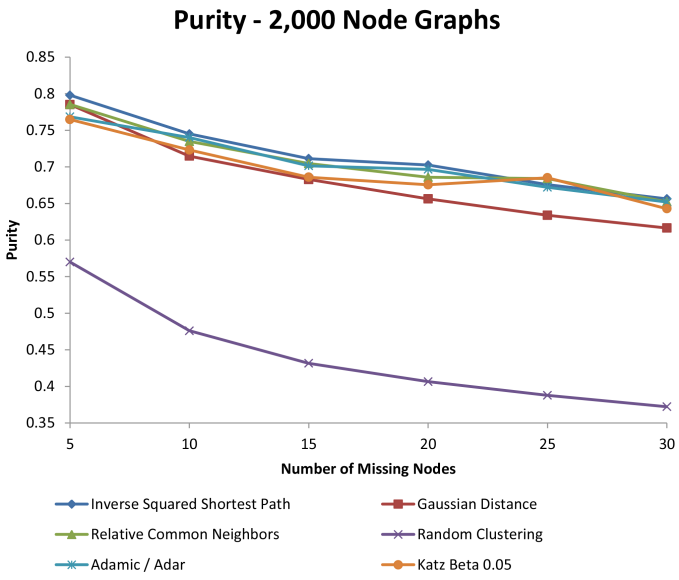

Fig. 5: Purity achieved when using each of the affinity measures.

It is important to note that subgraphs are very sparse, as each node has very few links on average. As a result, removing random nodes can cause the graph to become disconnected. Given this, it is interesting to see the performance of the spectral clustering algorithm on these subgraphs given in Figure 6. We can see that spectral clustering is effective also on these subgraphs, achieving much higher purity than the random clustering. Among the affinity measures, Inverse Squared Shortest Path (ISSP) performed the worst in this case and Adamic/Adar performed the best. This may be due to the fact that disconnected components lead to paths of infinite length, and ISSP is more sensitive to that.

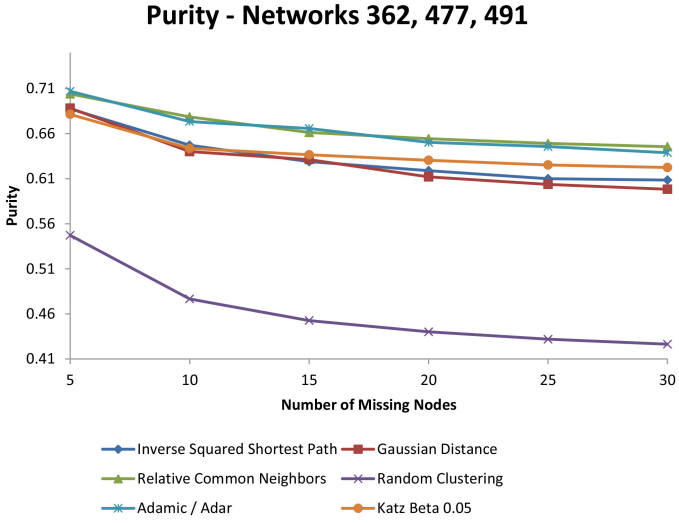

Fig. 6: Purity achieved when using each of the affinity measures on user network graphs.

As the key contribution of this paper lies within its use of affinity measures to solve the missing node problem, we considered whether a second clustering algorithm, k-means, could be used instead of spectral clustering. Overall, we found that the k-means algorithm also produced similar results, yet the spectral clustering algorithm overall performed significantly better. To evaluate this claim, we used 10 samples of 10,000 node networks. From each network, we randomly removed a set of missing nodes of sizes 30, 50, 100, 150 and 200. Then, we considered how the k-means

clustering algorithm could be used instead. Within both the spectral clustering and k-means algorithms, affinity matrices are needed to provide the input for the clustering algorithms. However, we considered two variations for which nodes to process the affinity measures within the k-means algorithms. Within the first variation, we considered both the known network nodes and the placeholder nodes. Within the second variation, we only processed the placeholder nodes. We randomly generated 8 networks, and for each network setting generated 10 sets of missing nodes. Thus, each datapoint in Figure 7 is averaged from 80 different experiments. As can be seen in this Figure, we considered three different affinity measures: Relative Common Neighbors (far left of the figure), Adamic Adar (middle of the figure) and Katz Beta 0.05 (far right of the figure). Note that for all three affinity measures, the spectral clustering algorithm performed better than both of the k-means variations. We also checked for normal distribution through the Shapiro-Wilks test. We found that the results for the k-means algorithms, with the exception of the 200 missing nodes case for the Adamic Adar and Katz Beta affinity measure, were normally distributed. We then performed the t-test to validate the results' significance. We found that the spectral clustering algorithm performed significantly better than both of the k-means variations. In all cases the p-scores were much lower than the 0.05 significance threshold. The largest p-score value was observed in the case of the related common neighbor affinity measure, where the p-scores between the spectral clustering and both k-means variations were much less than 0.0001.

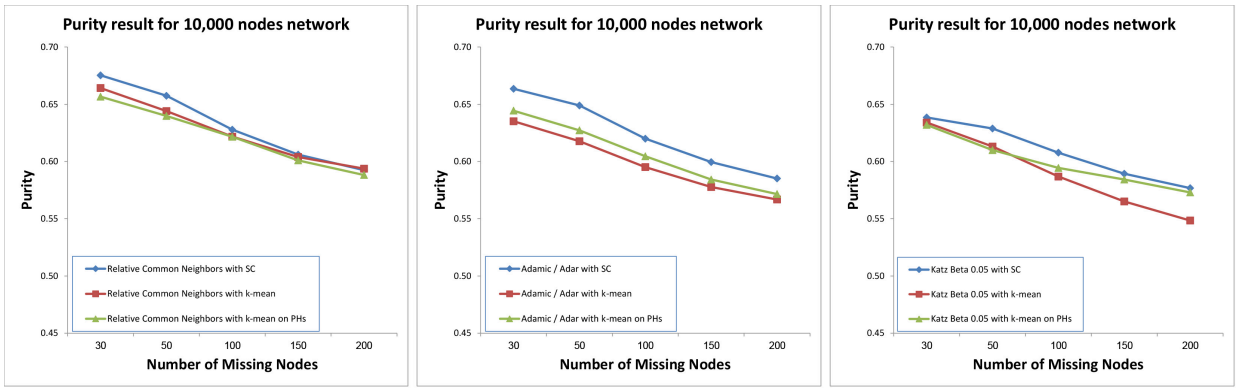

Fig. 7: Comparing the purity achieved through the spectral clustering algorithm (SC datapoints in the figure) versus the k-means clustering algorithm for the network (k-mean datapoints in the figure) and the missing nodes and k-means clustering on only the placeholders (k-means on PHs datapoints in the figure).

7. SCALING TO LARGE NETWORKS

Using a straightforward implementation of our proposed algorithm, we have been able to analyze network graphs of up to 5,000 nodes within a tractable time. This is typically not enough to deal with recent datasets of social networks, which can reach hundreds of thousands of nodes or more. Processing these datasets imposes serious memory and time constraints, and therefore efficient algorithms need to be developed and used for this task. In this section we propose two approaches to address this challenge by enabling efficient calculation of the affinity matrix and of the spectral clustering algorithm. We accomplish this through two novel extensions of the base MISC algorithm – adding Sparse Matrix Representation and Dimension Reduction.

7.1. Sparse Matrix Representation and Algorithms

Social networks generally have a sparse nature, i.e. each node is connected to only a small fraction of nodes. Most of the possible links do not exist, and therefore $|E| \ll |V|^2$. A commonly used representation for a graph $G = \langle V, E \rangle$ is a matrix $G^{|V|\times|V|}$ where $G_{i,j} = 1$ if $(i, j) \in E$, and $G_{i,j} = 0$ otherwise. When representing a social network as an adjacency matrix, most of the entries of the matrix will be zero. Using sparse matrix representations can significantly reduce the memory load and help deal with a much larger amount of data. It is possible to maintain only the non-zero values in order to preserve memory and efficiently perform calculations. The sparse property can also be utilized to improve the time complexity of the affinity matrix calculation. However, the affinity matrix itself may not be sparse in some cases, depending on the affinity measure used. For instance, if the Inverse Squared Shortest Path measure is used, and the graph is fully connected, then each pair of nodes has a non-zero affinity value. In this section we detail the algorithms for calculating the affinity matrix and analyze them in terms of space and time complexity. We detail how the Adamic/Adar can be modified to maintain the sparseness of the affinity matrix. While we also considered

how to modify the Relative Common Neighbors and Adamic/Adar affinity measures, the application of the general approach presented in this section is straightforward. Nonetheless, we include how these measures can be modified in Appendix A so that the results presented in this paper can be replicated.

ALGORITHM 3: - Katz Beta Affinity
Input: $G_a = \langle V_a, E_a \rangle$ - the available part of the network
K - the maximal path length to consider
β - the damping factor
Output: $A \in \mathbb{R}^{|V_a| \times |V_a|}$ – affinity matrix indicating the pairwise affinity

1: $P \longleftarrow G_a$ – initialize P as the adjacency matrix of G_a
2: $A_1 \longleftarrow \beta \cdot P$ – initialize A as the adjacency matrix of G_a, damped by β
3: **for** k = 2 to K **do**
4: $\quad P \longleftarrow P \cdot G_a$ – count the number of paths of length k by matrix multiplication
5: $\quad A_k \longleftarrow A_{k-1} + \beta^k \cdot P$ – accumulate the number of paths, damped by β^k
6: **end for**
7: Return A_K

Algorithm 3 describes the Katz Beta affinity calculation. As before, we assume that the given network graph, from which the affinity matrix is calculated, is $G_a = (V_a, E_a)$. We assume that the matrix is given in a sparse representation, i.e. a list of neighbors is given for each node. Let d be the maximal degree of a node in the graph. In step 1, the matrix P, which contains the number of paths between each pair of nodes, is initialized. At this point it contains only the number of paths of length 1, and it is therefore equal to the adjacency matrix which defines G_a. In step 2, the affinity matrix is also initialized, at this point taking into account paths of length 1, and damped by β. In steps 3-6 we iteratively increase the maximal path length taken into account until we reach paths of length K. In step 4 P is updated in each iteration k to ${G_a}^k$, indicating the number of paths of length k between each pair of nodes. P is then accumulated into A_k after damping by β^k, in step 5.

Multiplication of sparse $n \times n$ matrices can be done in $O(mn)$, where m is the number of non-zero elements. In our case, $n = |V_a|$ and $m = O(|V_a| \cdot d)$ for the matrix G_a. In P, the number of non-zero elements changes in each iteration by a factor of up to d. Therefore, in iteration k, step 4 requires $O(|V_a|^2 d^k)$ operations. The time complexity of steps 3-6 and of the entire procedure is then $O\left(\sum_{k=1}^{K} |V_a|^2 d^k\right) = O(|V_a|^2 d^K)$.

7.2. Dimension Reduction

In the first step of the original MISC algorithm described above, we calculate the affinity matrix $A \in \mathbb{R}^{|V_a| \times |V_a|}$, representing the pairwise affinity of nodes in V_a. This matrix is then fed into the spectral clustering algorithm. Each row of A corresponds to one of the nodes, represented in a vector space. Each column corresponds to a single dimension in the space in which these nodes reside. In steps 2-4 of spectral clustering (Algorithm 1), the nodes are embedded in a Euclidean space $\mathbb{R}^N$. By selecting the N largest eigenvectors of L in step 3, MISC reduces the number of dimensions given to the k-means algorithm in step 5 from $|V_v|$ to N. However, steps 2 and 3 still remain computationally expensive, dealing with a large matrix with $|V_a|$ rows and columns.

We propose a variation of MISC that includes dimension reduction, R-MISC, which can reduce the inherent complexity within the MISC algorithm, allowing us to tractably solve the missing node problem in much larger networks. R-MISC reduces the size of the affinity matrix, greatly decreasing the computational complexity of steps 2 and 3 within the original MISC algorithm. Intuitively, when the number of placeholders is relatively small compared to the total size of the network, many network nodes are "distant" from all of the placeholders in the sense that they have no affinity or only a very small affinity with the placeholders. Since only the placeholders are clustered in our problem setting, we propose that the dimensions representing these nodes can be removed from the affinity matrix with only a small loss in performance of the spectral clustering algorithm, yet significantly reducing the time and the memory used by the spectral clustering component of the MISC algorithm. Note that while these nodes have a very small affinity with the placeholders, they may have a high affinity with other nodes which are not removed. Thus, removing them may still affect the result of the spectral clustering. Yet, as our results show, the effect is minimal.

To test this method, we conducted experiments comparing the performance of R-MISC relative to MISC. After calculating the affinity matrix common to both algorithms, we remove the rows and columns within R-MISC that correspond to nodes that have zero affinity with all of the placeholders, resulting in a much smaller, square affinity matrix which is then used in the following steps of the original MISC algorithm.

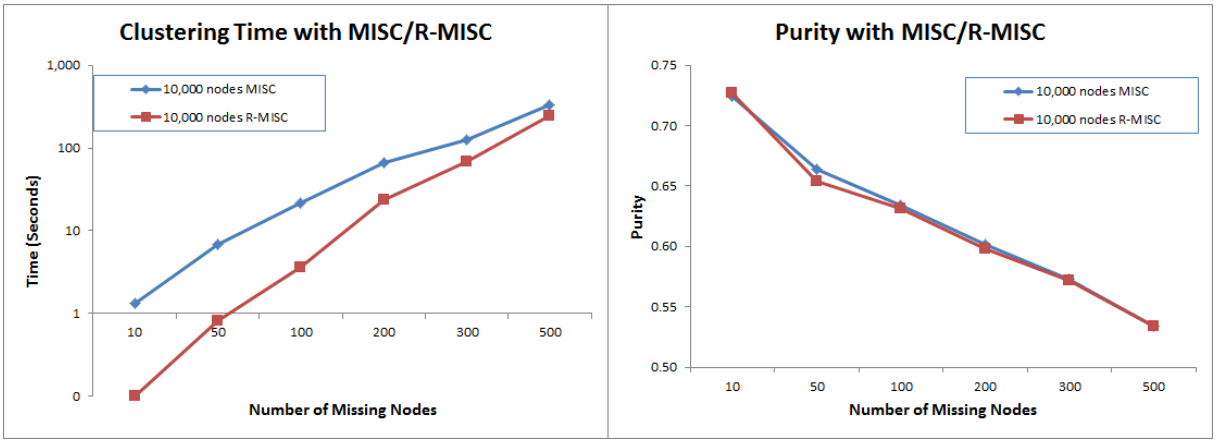

Fig. 8: The effect of dimension reduction on purity in 10,000 node graphs in terms of time (left) and purity (right) given different numbers of missing node sizes (X-axis in both sides).

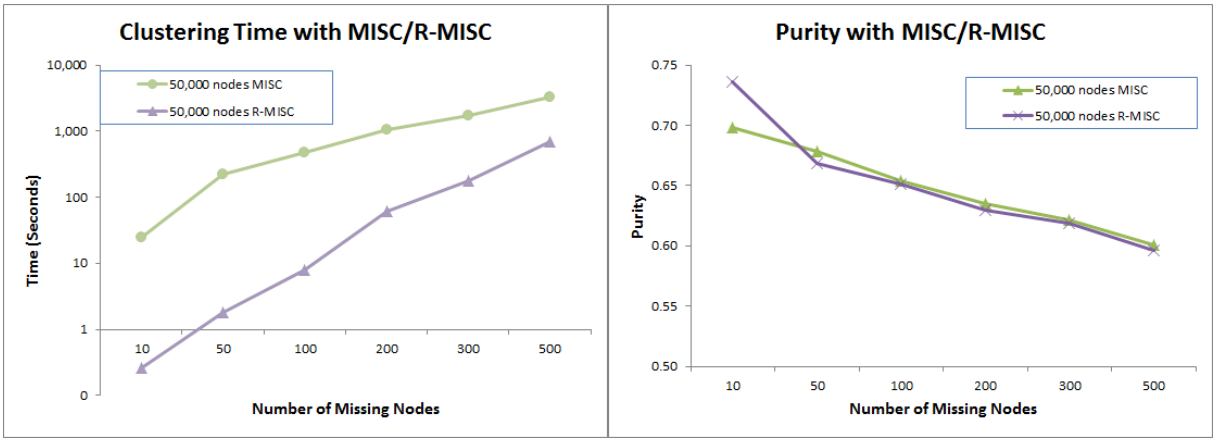

Fig. 9: The effect of dimension reduction on purity in 50,000 node graphs in terms of time (left) and purity (right) given different numbers of missing node sizes (X-axis in both sides).

We then studied the impact of employing dimension reduction within different network sizes and for different amounts of missing nodes within those networks. First, we studied how this approach is impacted by network size. In Figures 8 and 9 we present results from networks constructed with 10,000 and 50,000 nodes, respectively. We also studied additional networks with 20,000, 30,000 and 40,000 nodes and found R-MISC to be equally effective in these intermediate sizes as well. Second, we studied how the R-MISC would perform for different numbers of missing nodes. We studied networks with N = 10, 50, 100, 200, 300 and 500 missing nodes. Note that these values constitute the x-axis within Figures 8 and 9.

Last, we studied the impact that the network size and different missing node sizes had on performance. In doing so, we compared both the accuracy, as measured by purity, and the time, measured in seconds, for both the original MISC algorithm and the R-MISC variation with dimension reduction. The left side of Figures 8 and 9 represents the results from the time experiments, and the right side represents the corresponding purity measures. As expected, the dimension reduction significantly reduces the time needed to run the spectral clustering component of the algorithm, while only having a small impact on the R-MISC's performance. These graphs present results from the Relative Common Neighbors affinity measure. However, similar results were obtained when the other affinity measures were used. In all cases, the results clearly show that despite the overhead of performing the dimension reduction, significant time is saved and this method allows us to scale R-MISC to much larger networks within a tractable time. Note that as these experiments only pertain to the time used by the spectral clustering step of the algorithm, the time element of these results is not dependent on the affinity measure which we used. Thus, the time results are the same for all affinity measures being used.

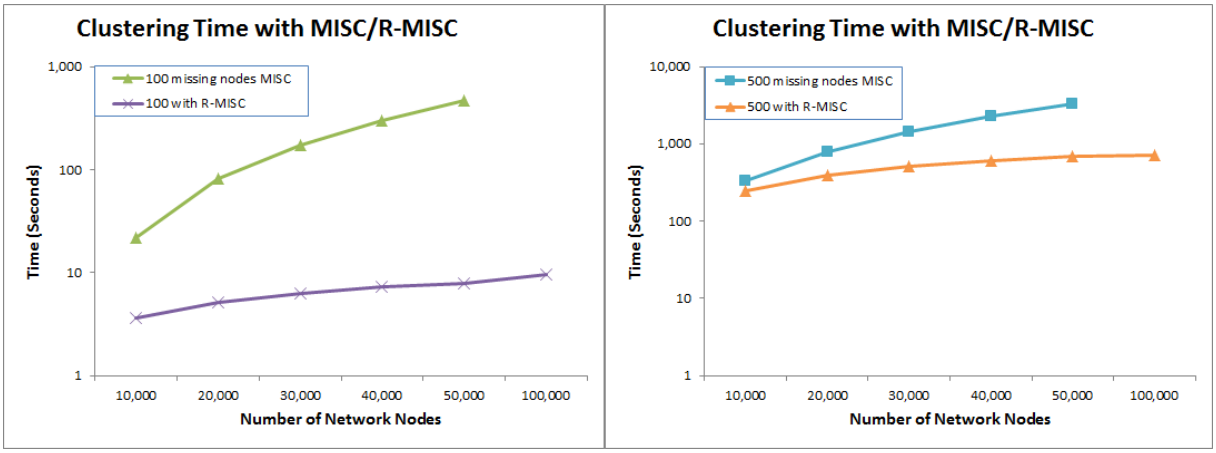

Fig. 10: The impact of network size on the MISC and R-MISC algorithms for networks with 100 (on left) and 500 (on right) missing nodes.

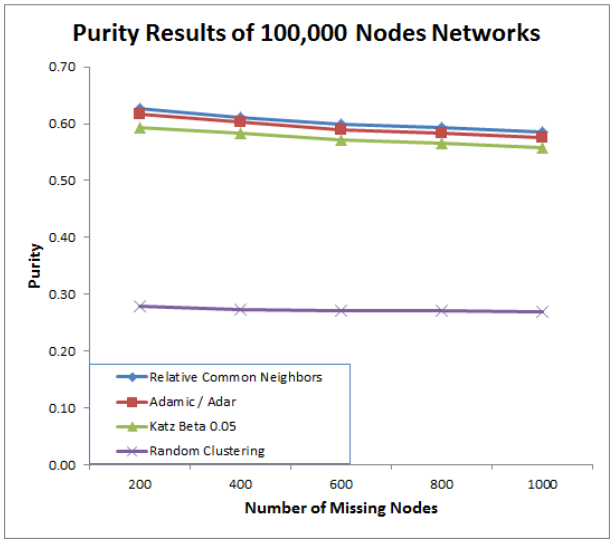

Fig. 11: Purity achieved by the R-MISC algorithm when using different affinity measures on 100,000 node graphs, using dimension reduction.

We also considered how scalable the R-MISC algorithm would be by considering networks with up to 100,000 nodes. We again considered networks with 100, 200, 300, 400 and 500 missing nodes. Figure 10 presents the results from the experiments with 100 and 500 missing nodes. Please note from these results that only the R-MISC algorithm was able to tractably solve the networks with 100,000 nodes. In comparison, note that time required by R-MISC algorithms increases relatively slowly with the network size. For example, within the 500 missing node problem, the time required for the R-MISC algorithm to solve for 500 missing nodes within a 30,000 node network is 504.30 seconds, while the time to solve for 500 nodes within 40,000, 50,000 and 100,000 nodes is 602.99, 683.77, and 702.83 seconds respectively. Thus, we conclude that R-MISC is a scalable algorithm, as can be extrapolated from these results.

Figure 11 displays the purity achieved by using three different affinity measures in the R-MISC algorithm, on graphs containing 100,000 nodes. The results are compared to a random clustering. The ability to analyze such large graphs is achieved due to dimension reduction and sparse matrix representation. The results of the tested measures are consistent with results from experiments on smaller graphs, with Adamic/Adar and Relative Common Neighbors performing slightly better than Katz Beta.

8. COMPARING MISC TO KRONEM

Recall from Section III that the *Missing Node Problem* is relatively new, with very few algorithms available to compare with MISC. Nonetheless, we did perform comparisons between MISC and the only other known algorithm suitable for the missing node problem, KronEM [Kim and Leskovec 2011]. KronEM accepts a known graph and the number of missing nodes and outputs a graph which predicts the missing nodes and their links. This algorithm is not based on the existence of placeholders and therefore it does not use them. Another key difference between MISC and KronEM is that KronEM assumes that the number of nodes in the full graph is a power of two.

Nonetheless, to facilitate a fair comparison between MISC and KronEM, we generated 10 networks, each containing $2^{11} = 2048$ nodes sampled from the full Facebook dataset. From each network, we randomly removed N missing nodes. The selected values for N were 11, 21, 31, 41 and 50 which are, respectively, approximately 0.5%, 1%, 1.5%, 2% and 2.5% of the network. The experiment was repeated four times for each selected value of N in each network, resulting in 40 iterations per missing node percentage. Each resulting network was fed to the KronEM and MISC algorithms.

The output of each algorithm was compared to the original network graph using Graph Edit Distance, since KronEM cannot be evaluated by Purity, as it is not a clustering based algorithm. As a baseline algorithm, we also measured the GED obtained by the random clustering approach in this set of experiments. An important observation is that MISC only predicts the existence of links between the predicted nodes and the neighbors of the original missing nodes. KronEM, on the other hand, might predict an erroneous link between a predicted node and any other node in the network, which was not connected to the missing node. This is due to the fact that KronEM does not take advantage of the placeholders. Therefore, in order to fairly compare the two, when calculating the GED, we only considered edit operations relating to links between the predicted nodes and the neighbors of the missing nodes. In addition, even though KronEM predicts a directed graph, we treated each predicted link as undirected, since this slightly improved the results of KronEM.

8.1. KronEM Configuration

The KronEM algorithm accepts the known network, without the placeholders, and the number of missing nodes. Additional parameters are the initial initiator matrix of the Kronecker graph, the set of parameters for the Gibbs sampling and the number of iterations [Kim and Leskovec 2011].

The authors recommended using at least 3 kinds of initial initiator matrices, which represent different relationships between the nodes. Accordingly, we ran the KronEM algorithm with the 3 recommended variations of the initial initiator matrix which represent the following relationship: (a) Homophily - love of the same; (b) Heterophily - love of the different; and (c) Core-periphery - links are most likely to form between members of the core and least likely to form between nodes members of the periphery. For all 3 variations we used the same parameters[1]. Each iteration took several hours to run using the C++ implementation we received from the authors of [Kim and Leskovec 2011].

8.2. MISC Configuration

For this comparison, we used MISC with three different affinity measures: Adamic/Adar, Katz Beta 0.05 and Relative Common Neighbors. Even though the tested network size was only 2048 nodes, we still used the dimension reduction and sparse network representation features of MISC, since they are shown to speed up the calculation with only a slight impact on the predictive performance. Each iteration of the algorithm ran for less than 10 seconds in Matlab[2].

8.3. Results

The results show that both MISC and KronEM algorithms are significantly better than the random clustering baseline. Between the three variations of KronEM, the results were almost identical, yet the Homophily variation produced slightly better results as the missing node percentage grew.

Between the three variations of affinity measures for the MISC algorithm, there was also a small difference, yet Relative Common Neighbors performed slightly better than Adamic/Adar and Katz Beta 0.05. All three algorithms performed significantly better than KronEM with the Homophily initiator matrix. The best obtained results from each algorithm are displayed in Figure 12. Table 1 displays the mean result and standard deviation achieved by each algorithm over the 40 iterations over each missing node percentage and four random draws of missing nodes from each of the 10 networks.

---

[1] In running the KronEM algorithm we set the algorithm parameters to the values as described in the ReadMe file in their algorithm's distribution. As such, the EM parameter which controls the number of iterations was set to 50, the l parameter controlling the gradient learning rate was 1e-5, the mns parameter controlling the minimum gradient step was 1e-4, and the mxs parameter controlling the maximum gradient step was 1e-2. The tests were run on a Windows Web Server 2008 with a 64 bit operation system, 4 GB RAM and an Intel dual core 2.27 GHz CPU.

[2] The MISC experiment ran on an Intel Core i5, 2.67 GHZ CPU with 4 GB of RAM, running Matlab 2010a and a Windows 7 OS.

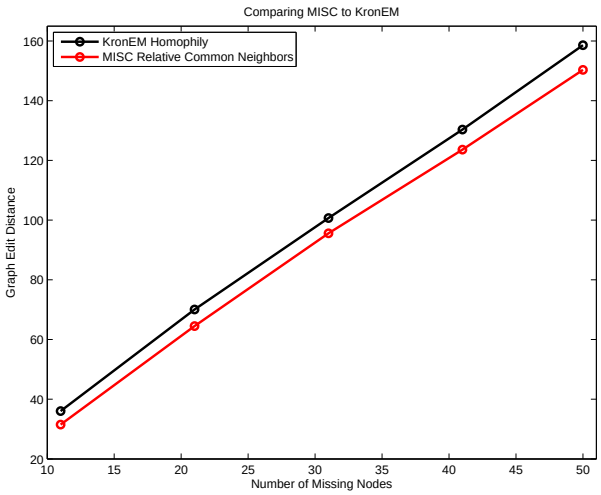

Fig. 12: Comparison of MISC to KronEM on 2,048 node networks using Graph Edit Distance (lower is better).

To determine the significance of these results, we first used the Shapiro-Wilk test to determine if the data was normally distributed. We found that the results of many of the networks that were studied were in fact not normally distributed. As a result, we used the Wilcoxon Sign-Rank Test to see if the differences in Table 1 were found to be statistically significant. Based on this test, we found that the differences were significant in all cases.

We conclude that while KronEM can be an excellent solution when a large part of the network is unknown and there are no indications of missing nodes, MISC can perform better when the placeholders are given and when trying to identify a small number of missing nodes.

9. DEALING WITH PARTIAL AND INACCURATE PLACEHOLDER INFORMATION

The algorithms and experiments described in this work have made assumptions regarding the information that is available on the social network. For instance, the number of missing nodes was assumed to be given as an input to the algorithm. In addition, it was assumed that all the placeholders can be identified with complete certainty. When a node was removed from the original graph in the experiments described above, a placeholder was attached to each one of that node's links. If we assume that indications of missing nodes in the network are obtained from a data mining module, such as image recognition and text processing, it is likely that this information would be partial and noisy. For instance, the number of missing nodes may not be known, not all placeholders may be known, and there may be false alarms indicating placeholders which do not actually exist. In this section we present methods aimed at addressing these issues.

Specifically, in this section we consider three different types of placeholder uncertainty. In the first type of case, we do not know the exact number of missing nodes. In this case, we must estimate this value according to the estimates we present. We consider a second case where insufficient placeholders exist to correctly identify all missing nodes. Last, we consider the case where extra placeholders exist. In this case, we assume that the actual missing nodes are found within the set of placeholders, but extraneous information exists that assumes additional placeholders exist which do not correspond to actual nodes within the network.

For the first two categories of problems (unknown number of missing nodes and missing placeholders) we use evaluations based on Graph Edit Distance, as the actual number of missing nodes is unknown based on the data. In these problems, the purity measure is not appropriate as the actual number of missing nodes is needed to calculate the purity measure. In contrast, Graph Edit Distance is based on calculating the number of transformations between the original and actual networks– something that can be calculated even without knowing the number of missing nodes. In contrast, in the last type of uncertainty, where there are extra placeholders, we again assume that the number of placeholders is known, allowing us to consider the purity evaluation measure as well.

9.1. Estimating the Number of Missing Nodes

In many real-life scenarios, a partial network may be available when the number of missing nodes is unknown. In the MISC algorithm, the number of clusters is set to equal the number of missing nodes, N. Recall that the placeholders in each cluster are united to one predicted node which is connected to all of the neighbors of these placeholders. Each

Table I: Comparison of MISC and KronEM Algorithms

Missing Node Percentage	Algorithm	Mean GED	Std.
0.5%	KronEM Core-Periphery	36.32	11.98
	KronEM Heterophily	36.18	12.16
	KronEM Homophily	36.03	11.16
	MISC Adamic/Adar	32.55	15.44
	MISC Katz Beta 0.05	31.03	16.00
	MISC CommNeighbors	31.52	16.47
1%	KronEM Core-Periphery	69.72	19.57
	KronEM Heterophily	70.20	20.82
	KronEM Homophily	70.08	19.39
	MISC Adamic/Adar	67.80	26.83
	MISC Katz Beta 0.05	66.45	28.76
	MISC CommNeighbors	64.50	27.57
1.5%	KronEM Core-Periphery	101.82	23.68
	KronEM Heterophily	102.63	27.08
	KronEM Homophily	100.70	23.92
	MISC Adamic/Adar	99.68	33.47
	MISC Katz Beta 0.05	97.65	33.15
	MISC CommNeighbors	95.60	32.99
2%	KronEM Core-Periphery	133.20	28.08
	KronEM Heterophily	134.75	29.71
	KronEM Homophily	130.33	26.74
	MISC Adamic/Adar	127.67	37.42
	MISC Katz Beta 0.05	127.05	37.26
	MISC CommNeighbors	123.62	35.92
2.5%	KronEM Core-Periphery	162.98	30.98
	KronEM Heterophily	167.25	36.26
	KronEM Homophily	158.60	29.39
	MISC Adamic/Adar	154.73	37.42
	MISC Katz Beta 0.05	155.23	40.00
	MISC CommNeighbors	150.35	39.66

Table II: Studying the Accuracy of the Estimation for the Number of Missing Nodes

	10	**20**	**30**	**50**	**70**	**100**	**Total**
Quadratic Degree Estimation	0.1850	0.1725	0.1308	0.1115	0.0921	0.0728	**0.1275**
Mean Degree Estimation	0.1850	0.1725	0.1300	0.1125	0.0921	0.0743	**0.1277**

predicted node is supposed to resemble one of the missing nodes. When the number of missing nodes is unknown, it must be estimated so that the predicted graph is as similar as possible to the original graph.

Our general approach is to estimate the number of missing nodes based on the general structure of social networks. We assume that a set ratio exists between nodes within the network and their neighbors. This relationship has been previously studied within missing link and node literature [Gomez-Rodriguez et al. 2012; Lin et al. 2012; Kim and Leskovec 2011]. We considered two methods, *quadratic* and *mean* estimations. Within the quadratic estimation we assumed that the number of edges in the network follows a quadratic relation of the network's nodes. This was also observed in the work of Kim and Leskovec [Kim and Leskovec 2011]. Accordingly, we first calculated this ratio according to the known part of the network, i.e. $|E_k| = a * |V_k|^2$, and we can then calculate the estimated number of clusters, $\hat{N}$, as the solution for the following equation: $|E_k \cup E_p|$ = a* $(|V_k| + N)^2$. The estimated $\hat{N}$ is rounded to the nearest integer and used in the MISC algorithm. Within the mean estimation we assigned d as the average degree of a node in V_a. The expected number of clusters is then $|V_p| / d$. The estimated number of clusters, $\hat{N}$, is rounded to the nearest integer and used in the MISC algorithm. This estimation causes $|V_p|$ nodes to be clustered into $\hat{N}$ clusters, and the average number of placeholders per cluster is then $|V_p| / \hat{N}$. As a result, the predicted nodes have the same average degree as the nodes in the available graph V_a. This is due to the fact that each placeholder has one link, and each

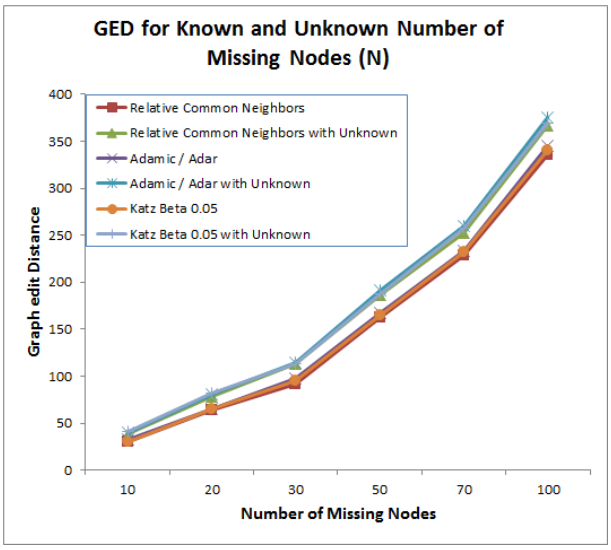

Fig. 13: Graph Edit Distance achieved when N is estimated and when N is known in advance. Tested on 10,000 node graphs.

predicted node will have the links of all the placeholders which were clustered into the relevant cluster representing that missing node.

We first studied how accurate these estimations were in predicting the actual number of missing nodes within the network. To do so, we ran an experiment using 10 samples of 10,000 node networks. From each network, we randomly removed a set of missing nodes of sizes 10, 20, 30, 50, 70 and 100. We calculated the two estimations according to the remaining networks and the placeholders. We then calculated the mean absolute error between each estimation and the actual number of missing nodes, and repeated this test 4 times. The results in Table II show the mean absolute error for these 40 runs. Please note that both estimation methods yielded similar results and that as the number of missing nodes increases, then the error is reduced. This implies that both estimations actually become more precise as more nodes are missing. One possible explanation for this is that as the number of missing nodes increases, small variations within localized portions of the network become less significant, making these estimations more accurate. Next, to measure the performance of using only an estimate of the number of missing nodes, we considered the predictive performance of the MISC algorithm when the actual number of clusters is given, in comparison to the performance obtained when using an estimate. In this case, the Graph Edit Distance measure was used since each instance of the experiment may result in a different number of clusters, depending on the selected subset of the network and the randomly chosen missing nodes. As both the *mean* and *quadratic* estimations yield similar results, Figure 13 displays the results of these experiments when using the *mean* estimate. While there is some visible degradation of the results achieved when N was unknown, the algorithm was still able to achieve beneficial results. For instance, the proposed algorithm which estimated the value of N still performed better than a random clustering algorithm which was given the correct value in advance, especially as N increased. A possible reason for some degradation in the results might be a large variance in the degrees of the missing nodes, causing a large variance in $|V_p|$, because each neighbor of a missing node is connected to a placeholder in the experiment. Thus, when a node with a large degree was randomly selected for removal, this led to an overestimate of N. The estimation of N is expected to be much more precise if the subset of the network is selected in a way that the variance in the nodes' degrees would be smaller.

9.2. Addressing Missing Placeholders

Many real-life scenarios are likely when some of the placeholders are missing, for example, if the indications for some of the placeholders were not obtained. Under the assumption that a data mining module provides us with the indications of the existence of placeholders, this module may provide incomplete results. This mistaken information will not only add false negatives of non-detection of placeholders, but false positives may also exist in the form of false alarms. To assess how robust MISC is with incomplete placeholder information, we measured how this partial information would affect the missing nodes' identification. For this purpose we conducted a set of experiments where the percentage of known placeholders ranged from 10% to 100% of the total placeholders that are created when removing the missing nodes. Figure 14 displays the Graph Edit Distance achieved by the algorithm for each percentage of known placeholders. The affinity measures used in these experiments were Katz Beta and Adamic/Adar. The results

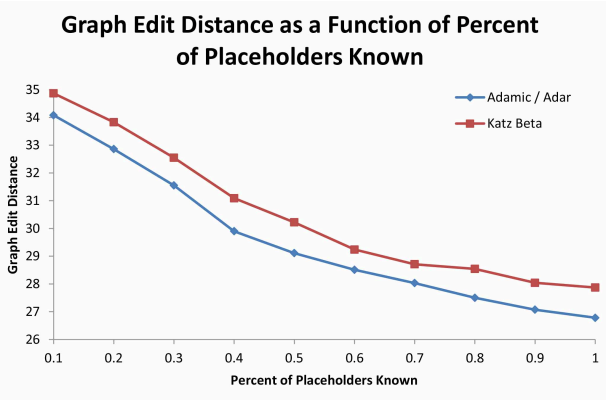

Fig. 14: Graph Edit Distance between the original graph and the graph predicted by the MISC algorithm when only some of the placeholders are known.

clearly show that the performance decreases when less placeholders are known, resulting in a higher Graph Edit Distance between the original graph G and the predicted graph $\hat{G}$. Each point in the graph represents the average Graph Edit Distance achieved from 100 experiments. This scenario raises questions regarding the MISC algorithm's ability to address missing information within the placeholders. First, when some placeholders are unknown, the resulting graph predicted by the spectral clustering algorithm would lack the edges between each unknown placeholder and its neighbor. In addition, adapting the spectral clustering algorithm to deal with probabilistic placeholders is not trivial. A possible approach is to run a Link Prediction Algorithm in order to complete the edges which are missed due to the unknown placeholders. To formulate this new problem we alter the input of the original missing node identification problem. Recall that V_p is the group of all of the placeholders generated from the missing nodes and their edges. We define the following new groups: V_p^k, E_p^k - the group of known placeholders and their associated edges, and V_p^u, E_p^u - the group of unknown placeholders and their associated edges, such that $V_p = V_p^k \cup V_p^u$ and $V_p^k \cap V_p^u = \phi$. For each missing node $v \in V_m$, and for each edge $\langle v, u \rangle \in$ E, $u \in V_k$, we add a placeholder v' either to V_p^k or to V_p^u and an edge $\langle v', u \rangle$ to E_p^k or to E_p^u accordingly. The available network graph, $G_a = \langle V_a, E_a \rangle$, now consists of $V_a = V_k \cup V_p^k$ and $E_a = E_k \cup E_p^k$. In addition, we define the indicator function $I(v)$ which returns the value 1 for each known node $v \in V_k$ if it is connected to an unknown placeholder, and otherwise returns 0.

$$I(v) = \begin{cases} 1 & if \ \exists u \in V_p^u \ \wedge \ \langle v, u \rangle \in E_p^u \\ 0 & otherwise \end{cases}$$

The value of $I(v)$ is of course unknown to the system. Instead, we model the data mining module's knowledge as $S(v)$, a noisy view of $I(v)$ with additive random noise $X(v)$, where X is an unknown random variable.

$$S(v) = I(v) + X(v)$$

The formal problem definition for this problem setting is then: given a known network $G_k = \langle V_k, E_k \rangle$, an available network $G_a = \langle V_a, E_a \rangle$, the value of $S(v)$ for each $v \in V_k$ and the number of missing nodes N, divide the nodes of $V_a \backslash V_k$ to N disjoint sets $V_{v_1}, \ldots, V_{v_N}$ such that $V_{v_i} \subseteq V_p$ are all the placeholders of $v_i \in V_m$, and connect each set to additional nodes in V_k such that the resulting graph has a minimal graph edit distance (GED) from the original network G. We propose two possible algorithms for solving this problem – *Missing Link Completion* and *Speculative MISC*. Both algorithms accept a threshold T, which indicates a minimal value of $S(v)$ to be taken into account as a valid indication of a placeholder. The *Missing Link Completion* algorithm at first ignores $S(v)$, and runs the original *MISC* algorithm only on the available graph. Then, in order to complete links that are missing from the predicted graph due to not knowing all the placeholders, a Link Prediction algorithm is used. This algorithm simply connects each node that has a value of $S(v)$ which is higher than T to the new node with which it has the strongest affinity. Algorithm 6 details the pseudo code of the *Missing Link Completion* algorithm: The *Speculative MISC* algorithm uses a different approach regarding $S(v)$. Under this approach, a new placeholder is added and connected to every node v whose indication value, $S(v)$ is greater than T. Next, the regular MISC algorithm (algorithm 2) is used on the new graph to predict the original graph. To compare these two algorithms, we conducted experiments where only 50% of the placeholders are known. For each node $v \in V_k$ we calculated $S(v)$ as described above, indicating if that node

ALGORITHM 4: - Missing Link Completion

Input:
$G_k = \langle V_k, E_k \rangle$ {the known part of the network}
$G_a = \langle V_a, E_a \rangle$ {the available part of the network}
$S(v)\,,\ \forall v \in V_k$ {the available indications about the unknown placeholders}
N {the number of missing nodes}
$\alpha : G(V, E) \longrightarrow \mathbb{R}^{|V|\times|V|}$ {a function for calculating the affinity matrix of nodes in a graph}
T {the minimal value of $S(v)$ to be taken into account as a valid indication of a placeholder}
Output: $\hat{G} = \left\langle \hat{V}, \hat{E} \right\rangle$ - prediction of the original network G

1: $\hat{G} = \left\langle \hat{V}, \hat{E} \right\rangle \longleftarrow$ ***MISC*** $(G_k,\ G_a, \text{N}, \alpha)$ {Apply the MISC algorithm while ignoring $S(v)$}
2: $\hat{A} = \alpha(\hat{G})$ {Calculate the pairwise affinity of the nodes in the predicted graph $\hat{G}$} {Iterate over all the known nodes with a value of $S(v)$ higher than the threshold}
3: **for** each $v \in V_k : S(v) > T$ **do**
4: $\quad u = \text{argmax}_{w \in \hat{V} \setminus V_k} \hat{A}(v, w)$ {find the new node u with the strongest affinity to node v}
5: $\quad \hat{E} \longleftarrow \hat{E} \cup \{\langle v, u \rangle\}$ {add a new edge between v and u}
6: **end for**
7: **return** $\hat{G} = \left\langle \hat{V}, \hat{E} \right\rangle$

ALGORITHM 5: Speculative MISC

Input:
$G_k = \langle V_k, E_k \rangle$ - the known part of the network
$G_a = \langle V_a, E_a \rangle$ - the available part of the network
$S(v)\,,\ \forall v \in V_k$ – the available indications about the unknown placeholders
N – the number of missing nodes
$\alpha : G(V, E) \longrightarrow \mathbb{R}^{|V|\times|V|}$ - a function for calculating the affinity matrix of nodes in a graph
T - minimal value of $S(v)$ to be taken into account as a valid indication of a placeholder
N - number of clusters
Output: $\hat{G} = \left\langle \hat{V}, \hat{E} \right\rangle$ - prediction of the original network G

1: **for** each $v_i \in V_k : S(v_i) > T$ **do**
2: $\quad V_a \longleftarrow V_a \cup \{p_i\}$ - add a new placeholder to V_a
3: $\quad E_a \longleftarrow E_a \cup \{\langle v_i, p_i \rangle\}$ - add an edge to the placeholder
4: **end for**
5: $\hat{G} = \left\langle \hat{V}, \hat{E} \right\rangle \longleftarrow$ ***MISC*** $(G_k,\ G_a, \text{N}, \alpha)$
6: Return $\hat{G} = \left\langle \hat{V}, \hat{E} \right\rangle$

is connected to an unknown placeholder, and added Gaussian noise with a mean of 0 and standard deviation of $1/4$. In each experiment we decreased the value of T, so more indications would be taken into account. We inputted these values into the algorithms, and measured the graph edit distance achieved by each algorithm. The results were also compared to a baseline algorithm which executed MISC while ignoring the additional indications in $S(v)$ – Algorithm 2 from above. Figure 15 shows the graph edit distance achieved by each method as a function of the number of indications taken into account. Each method used Adamic/Adar as the affinity measure in this set of experiments, since this affinity measure showed good results both in the classic Link Prediction Problem, and in the missing node identification problem. We can see that when a small number of indications are used, and the probability of taking an invalid placeholder into account is low, both methods improve the graph edit distance in comparison to the baseline method, which ignores the indications. When more indications are taken into account the probability of error increases, causing the graph edit distance to increase because invalid placeholders are used. For any value of T, we can clearly see that *Speculative MISC* outperforms *Missing Link Completion* by achieving a lower graph edit distance. Figure 16 illustrates the same comparison when using the Katz Beta affinity measure. Each point in these two figures represents the average of 100 experiments. Figure 17 studies how the performance of MISC degrades as fewer placeholders are known relative to the KronEM algorithm [Kim and Leskovec 2011]. Within this result, we studied the impact of performance, as measured by Graph Edit Distance, and used the Relative Common Neighbors affinity matrix within

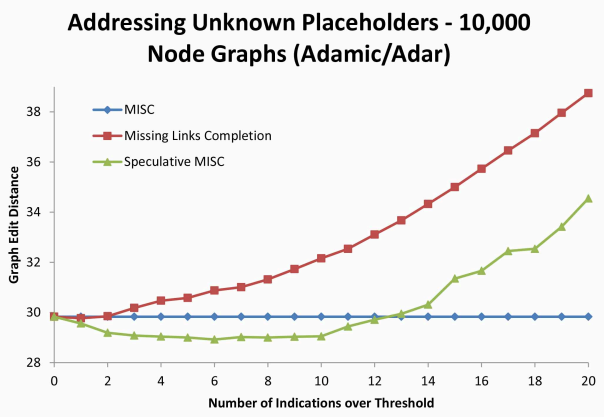

Fig. 15: Graph Edit Distance between the original network graph and the graph predicted by algorithms which attempt to address unknown placeholders, using Adamic/Adar affinity measure.

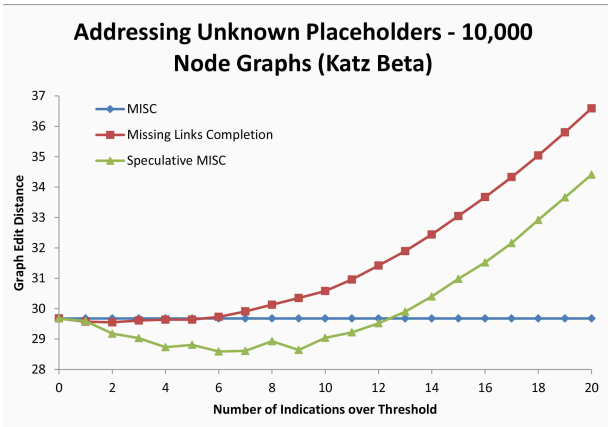

Fig. 16: Graph Edit Distance between the original network graph and the graph predicted by algorithms which attempt to address unknown placeholders, using Katz Beta affinity measure.

MISC. As per our results in Section VII, note that the MISC algorithm without any missing information performs significantly better than the KronEM algorithm. We then proceeded to remove information about the placeholders. We used the same 40 instances of the 10 samples of the 2048 node network with the same missing nodes we used within the original experiment from Section VII. We found that even with 70% of the placeholder information being missing (only 30% known), the results of the MISC algorithm were nearly identical to those of KronEM. Thus, we concluded that the MISC algorithm is superior to KronEM as long as a minimal amount (30%) of placeholder information is known.

9.3. Evaluating the Effect of Extraneous Placeholders

Within the last scenario we consider, we consider the impact having too many placeholders will have on the MISC algorithm. Within this possibility, we again assume that some noise exists within the number of placeholders, say again because of errors made by a data mining module to provide indications of possible missing nodes, only this time we consider that false positives may provide indications of placeholders when they in fact do not exist. To evaluate this possibility, we considered three methods of how to add the extraneous placeholders, one global method and two local methods. In the global method, we added extraneous placeholders which were randomly connected to the network nodes. The motivation behind this method is that we wished to ascertain the impact of random noise within the network. Thus, adding extraneous placeholders in this fashion replicates this type of noise. In the local methods, we added extraneous placeholders which were not randomly added, but were added with a uniform distribution throughout the network onto the nodes with existing placeholders. We added these extraneous placeholders at either a distance of 2 or 3 from the original true missing node (placeholder). For example, if a given node X has one placeholder,

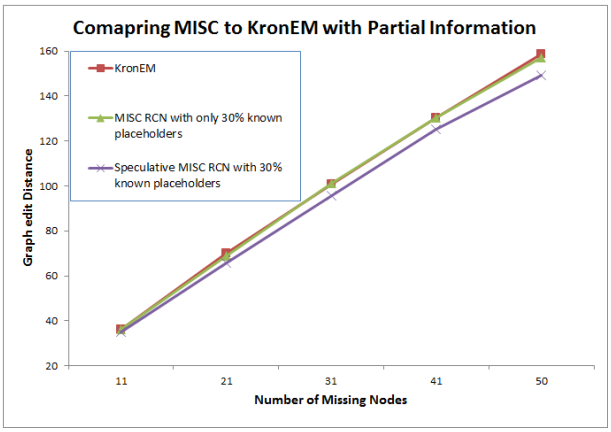

Fig. 17: Comparing the performance of KronEM with MISC, and MISC with only 30% of the placeholders being known.

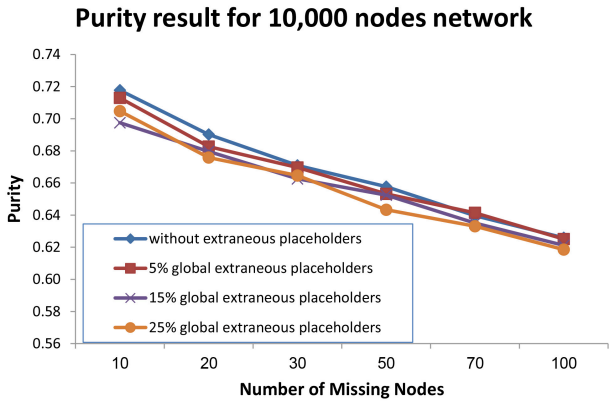

Fig. 18: Purity achieved by MISC when extraneous placeholders exist.

within the distance 2 method we would add an extraneous placeholder to Y, which is a visible neighbor of X. In this experiment, we use again the 10 samples of a 10,000 node network. From each network, we randomly removed a set of missing nodes of sizes 10, 20, 30, 50, 70 and 100. For each network we added extraneous placeholders of 5%, 15% and 25% of the number of true placeholders using the three methods described above: global, local with distance 2 and local with distance 3. As the number of missing nodes is known in this experiment, we use both purity and the Graph Edit Distance measures. We repeated this test 8 times. The results in Figure 18 and Table III show the mean purity for these 80 runs for these different setups. The results of this experiment clearly demonstrate that the extraneous placeholders have only a slight effect on MISC's results. For the purity results, seen in Figure 18 and Table III, note that both local types of extraneous placeholders had almost no impact, while even the global extraneous placeholders had only a slight degradation in the results. Similarly, we found that MISC was also minimally impacted by extraneous placeholders within the Graph Edit Distance measures. These results contrasts with the result in the previous section. Note that while having too few placeholders strongly degrades the results, having extraneous placeholders only slightly decreases MISC's performance.

10. CONCLUSIONS AND FUTURE WORK

In this research we have introduced and formally defined the *Missing Node Identification* problem, where a partial network graph is given with indications as to which nodes are neighbors of missing nodes. Under this problem setting, these indications, which are marked as placeholder nodes, must be grouped into disjoint groups, each group representing one of the missing nodes. We present the novel algorithm, *MISC (Missing node Identification by Spectral Clustering)*, as a solution to these problem. This algorithm is based on the well known spectral clustering algorithm [Ng et al. 2001], used in many other problem settings. To adapt this algorithm to the Missing Node Identification problem,

Number of Missing Nodes	10	20	30	50	70	100	Total
without extraneous placeholders	0.7178	0.6902	0.6710	0.6577	0.6396	0.6259	0.6670
5% global extraneous placeholders	0.7130	0.6827	0.6697	0.6532	0.6414	0.6251	0.6642
15% global extraneous placeholders	0.6975	0.6796	0.6626	0.6525	0.6350	0.6211	0.6580
25% global extraneous placeholders	0.7047	0.6758	0.6646	0.6433	0.6331	0.6186	0.6567
5% local extraneous placeholders (dist 2)	0.7130	0.6889	0.6724	0.6531	0.6402	0.6261	0.6656
15% local extraneous placeholders (dist 2)	0.7131	0.6788	0.6685	0.6527	0.6394	0.6219	0.6624
25% local extraneous placeholders (dist 2)	0.7103	0.6812	0.6713	0.6508	0.6400	0.6209	0.6624
5% local extraneous placeholders (dist 3)	0.7077	0.6821	0.6695	0.6540	0.6423	0.6265	0.6637
15% local extraneous placeholders (dist 3)	0.7151	0.6818	0.6688	0.6527	0.6392	0.6241	0.6636
25% local extraneous placeholders (dist 3)	0.7103	0.6827	0.6683	0.6478	0.6373	0.6213	0.6613

Fig. 19: A table of the purity achieved by MISC when extraneous placeholders exist.

the fundamental step of calculating the affinity matrix is altered in the original algorithm. Affinity measures from the Link Prediction Problem [Adamic and Adar 2003; Katz 1953; Liben-Nowell and Kleinberg 2007] are combined with spectral clustering. This combination is shown by a set of experiments on real world Facebook data [Gjoka et al. 2010] to be useful in solving the Missing Node Identification problem. Further experiments show that the MISC algorithm is able to achieve a better predictive performance than the state of the art KronEM algorithm[Kim and Leskovec 2011]. After solving and analyzing the solution to the basic problem setting, we also presented two possible cases where the base algorithm must be altered. In the first case, we considered how the algorithm must be extended to support large scale networks of hundreds of thousands of nodes. To do so we presented two possible alterations to the algorithm. First, we suggested utilizing the sparse nature of social networks to overcome memory and time constraints. Efficient algorithms for calculating the affinity matrix using different affinity measures while utilizing the sparse nature are detailed. Secondly, a dimension reduction method is detailed which significantly decreases the time complexity with the spectral clustering steps of MISC, while causing only a small reduction of its performance. In the second case, we consider how the base algorithm must be altered to address cases where only partial information exists about the social network. We present a simple, yet effective, method of estimating the number of missing nodes if this number is not known in advance. Despite the use of an approximation for this number, our experiments show only a small reduction in performance over the base algorithm which requires knowing in advance the exact number of missing nodes. Next, we consider how the base algorithm would potentially degrade if incorrect information exists about missing nodes. In real life scenarios, a data mining module which provides the indications of the missing nodes in the known network should be assumed to be imperfect. There may be misdetections and false alarms when trying to automatically identify the indications. This advanced problem setting is also modeled and two solutions are proposed - *Missing Link Completion* and *Speculative MISC*. Speculative MISC is shown to be effective in improving the prediction quality when the number of false alarms is low. The two proposed solutions also enable a comparison between the MISC algorithm and classic Link Prediction algorithms, showing that MISC performs better.

There are many possible approaches to further develop this research. In our study, we solve the missing node problem exclusively through using the network structure and without using any information about nodes' or links' properties. We have begun to explore how to add properties about nodes' and links' attributes to better solve this problem. One possibility is to use additional node information based on the homophily principle (love of the same) that was previously studied [McPherson et al. 2001]. Accordingly to this principle, people's personal networks are homogeneous with regard to many sociodemographic, behavioral, and interpersonal characteristics. While the affinity measures within this paper implicitly contain elements of this principle, future work is necessary to study how additional information can and should be explicitly added.

An addition extension of this work could be defining and utilizing more complex indications of missing nodes. In the current research we model each indication as a placeholder, but other models may hold more information on each indication of a missing node. Another possible direction is dealing with datasets of millions of nodes by expanding the algorithm to utilize databases and distributed systems' infrastructure. A related area for future work could be developing data mining modules which analyze social network profiles and produce indications of missing nodes. These data mining modules could utilize natural language processing and computer vision algorithms to identify unrecognized people which are connected to the known members of the network. Additionally, if these data mining modules could also be used to infer if two placeholders (e.g., two images) refer to the same node (e.g., the same person). Thus, this information could possibly be used to rule in or out possible nodes for the placeholders. This direction also

assumes that specific information exists about a given node or link, above and beyond the general structure of the network, and is also something that we are actively studying.

REFERENCES

Lada A. Adamic and Eytan Adar. 2003. Friends and Neighbors on the Web. *Social Networks* 25, 3 (2003), 211–230.

Anat Almog, Jacob Goldberger, and Yuval Shavitt. 2008. Unifying Unknown Nodes in the Internet Graph Using Semisupervised Spectral Clustering. *Data Mining Workshops, International Conference on* (2008), 174–183.

Horst Bunke and Bruno T. Messmer. 1993. Similarity Measures for Structured Representations. In *EWCBR*. 106–118.

Aaron Clauset, Cristopher Moore, and M. E. J. Newman. 2008. Hierarchical structure and the prediction of missing links in networks. *Nature* 453, 7191 (May 2008), 98–101.

Motahhare Eslami, Hamid R. Rabiee, and Mostafa Salehi. 2011. DNE: A Method for Extracting Cascaded Diffusion Networks from Social Networks. In *SocialCom/PASSAT*. 41–48.

Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3-5 (2010), 75 – 174.

Antonino Freno, Gemma Garriga, C., and Mikaela Keller. Learning to Recommend Links using Graph Structure and Node Content. In *Neural Information Processing Systems Workshop on Choice Models and Preference Learning*. http://hal.inria.fr/hal-00641419

Minas Gjoka, Maciej Kurant, Carter T. Butts, and Athina Markopoulou. 2010. Walking in Facebook: A Case Study of Unbiased Sampling of OSNs. In *Proceedings of IEEE INFOCOM '10*. San Diego, CA.

Manuel Gomez-Rodriguez, Jure Leskovec, and Andreas Krause. 2012. Inferring Networks of Diffusion and Influence. *TKDD* 5, 4 (2012), 21.

Neil Zhenqiang Gong, Ameet Talwalkar, Lester W. Mackey, Ling Huang, Eui Chul Richard Shin, Emil Stefanov, Elaine Shi, and Dawn Song. 2011. Predicting Links and Inferring Attributes using a Social-Attribute Network (SAN). *CoRR* abs/1112.3265 (2011).

Roger Guimerà and Marta Sales-Pardo. 2009. Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences* 106, 52 (2009), 22073–22078.

Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18, 1 (March 1953), 39–43.

Myunghwan Kim and Jure Leskovec. 2011. The Network Completion Problem: Inferring Missing Nodes and Edges in Networks. In *SDM*. 47–58.

Gueorgi Kossinets. 2003. Effects of missing data in social networks. *Social Networks* 28 (2003), 247–268.

Orestis Kostakis, Joris Kinable, Hamed Mahmoudi, and Kimmo Mustonen. 2011. Improved call graph comparison using simulated annealing. In *Proceedings of the 2011 ACM Symposium on Applied Computing (SAC '11)*. 1516–1523.

Vincent Leroy, B. Barla Cambazoglu, and Francesco Bonchi. 2010. Cold Start Link Prediction. *SIGKDD 2010* (2010).

David Liben-Nowell and Jon Kleinberg. 2007. The Link-Prediction Problem for Social Networks. *Journal of the American Society for Information Science and Technology* 58, 7 (May 2007), 1019–1031.

Wangqun Lin, Xiangnan Kong, Philip S. Yu, Quanyuan Wu, Yan Jia, and Chuan Li. 2012. Community detection in incomplete information networks. In *WWW*. 341–350.

Miller McPherson, Lynn S. Lovin, and James M. Cook. 2001. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology* 27, 1 (2001), 415–444.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*. MIT Press, 849–856.

Mason A. Porter, Jukka-Pekka Onnela, and Peter J. Mucha. 2009. Communities in Networks. *Notices of the American Mathematical Society* 56, 9 (2009), 1082–1097.

Eldar Sadikov, Montserrat Medina, Jure Leskovec, and Hector Garcia-Molina. 2011. Correcting for missing data in information cascades. In *WSDM*. 55–64.

Alexander Strehl and Joydeep Ghosh. 2003. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.* 3 (March 2003), 583–617.

Ulrike von Luxburg. 2007. A Tutorial on Spectral Clustering. *Statistics and Computing* 17, 4 (December 2007), 395–416.

In Section VII, we presented how the Katz Beta affinity measure can be modified to be used on sparse matrix representations. Similarly, other affinity measures, such as the Relative Common Neighbors and Adamic Adar measures can also be applied. We present the exact algorithms used to create these modified affinity measures so that the results in the paper can be replicated. As was the case in Section VII, we assume the given network graph, from which the affinity matrix is calculated, is $G_a = (V_a, E_a)$ and that the matrix is given in a sparse representation, i.e. a list of neighbors is given for each node. Let d be the maximal degree of a node in the graph. Algorithm 6 describes the calculation of the affinity matrix using the Relative Common Neighbors affinity measure. In steps 2-6 of the algorithm, each placeholder is connected to the neighbors of its single neighbor. In steps 7-13, we iterate over each node and accumulate the affinity to each node with which the current node has a common neighbor. This is done efficiently by only going over the lists of neighbors in the sparse graph representation, instead of going over all of the possible nodes. Each update to $A_{i,j}$ is done in $O(d)$ since A is also a sparse matrix which does not enable random access in $O(1)$. This is because the appropriate entry has to be found in a list of length at most d during each update. Overall this loop (steps 7-13) requires a total of $O(|V_a| \cdot d^3)$ operations. Since this step is the bottleneck of the procedure, we conclude that the time complexity of the whole procedure is $O(|V_a| \cdot d^3)$. This is a significant improvement over the naive implementation using a non-sparse matrix, which would require $O(|V_a|^2)$ operations since $d \ll |V_a|$.

Algorithm 7 describes the affinity calculation using the Adamic/Adar affinity measure. As in the previous procedure,

ALGORITHM 6: - Relative Common Neighbors Affinity

Input: $G_a = \langle V_a, E_a \rangle$ - the available part of the network
$G_k = \langle V_k, E_k \rangle$ - the known part of the network
Output: $A \in \mathbb{R}^{|V_a| \times |V_a|}$ – affinity matrix indicating the pairwise affinity

1: $A \longleftarrow 0$ – initialize A to an all zero matrix
2: **for** each placeholder node $v \in V_a \backslash V_k$ and edge $\langle u, v \rangle$, **do**
3: **for** each neighbor $w \in A(u)$ **do**
4: $E_a \longleftarrow E_a \cup \{\langle v, w \rangle\}$ - connect the placeholder to its neighbor's neighbors
5: **end for**
6: **end for**
7: **for** each node $i \in V_a$ **do**
8: **for** each node $k \in \Gamma(i)$ **do**
9: **for** each node $j \in \Gamma(k)$ **do**
10: $A_{i,j} \longleftarrow A_{i,j} + 1$ (k is a common neighbor of i and j)
11: **end for**
12: **end for**
13: **end for**
14: **for** each pair $(i, j) \in A$ do: **do**
15: $A_{i,j} \longleftarrow A_{i,j} / min(|\Gamma(i)|, |\Gamma(j)|)$
16: **end for**

ALGORITHM 7: - Adamic/Adar Affinity

Input: $G_a = \langle V_a, E_a \rangle$ - the available part of the network
$G_k = \langle V_k, E_k \rangle$ - the known part of the network
Output: $A \in \mathbb{R}^{|V_a| \times |V_a|}$ – affinity matrix indicating the pairwise affinity

1: $A \longleftarrow 0$ – initialize A to an all zero matrix
2: **for** each placeholder node $v \in V_a \backslash V_k$ and edge $\langle u, v \rangle$ **do**
3: **for** each neighbor w of node u **do**
4: $E_a \longleftarrow E_a \cup \{\langle v, w \rangle\}$ - connect the placeholder to its neighbor's neighbors
5: **end for**
6: **end for**
7: **for** each node $i \in V_a$ **do**
8: **for** each node $k \in \Gamma(i)$ **do**
9: **for** ach node $j \in \Gamma(k)$ **do**
10: $A_{i,j} \longleftarrow A_{i,j} + 1/log(|\Gamma(k)|)$ (k is a common neighbor of i and j)
11: **end for**
12: **end for**
13: **end for**

here we also connect each placeholder to the neighbors of its single neighbor (steps 2-6). The next loop (steps 7-13) is very similar to the previous procedure, except that here the accumulated value in each iteration is $1/log(|(k)|)$. Since this value is calculated in constant time, the total time complexity of step 3 and of the entire procedure is also $O(|V_a| \cdot d^3)$, as in the previous procedure.