SAMI: An Algorithm for Solving the Missing Node Problem using Structure and Attribute Information

Sigal Sina, Avi Rosenfeld, and Sarit Kraus

Abstract—An important area of social network research is identifying missing information which is not visible or explicitly represented in the network. Recently, the *Missing Node Identification* problem was introduced where missing members in the social network structure must be identified. However, previous works did not consider the possibility that information about specific users (nodes) within the network may be known and could be useful in solving this problem. Assuming such information such as user demographic information and users' historical behavior in the network is known, more effective algorithms for the *Missing Node Identification* problem could potentially be developed. In this paper, we present three algorithms, `SAMI-A`, `SAMI-C` and `SAMI-N`, which leverage this type of information in order to perform significantly better than previous missing node algorithms. However, as each of these algorithms and the parameters within these algorithms often perform better in specific problem instances, a mechanism is needed to select the best algorithm and the best variation within that algorithm. Towards this challenge, we also present `OASCA`, a novel online selection algorithm. We present results that detail the success of the algorithms presented within this paper.

Index Terms—Algorithms, Social networks, K-means, missing nodes.

1 Introduction

Social networks, which enable people to share information and interact with each other, have become a key Internet application in recent years. These networks are typically represented as graphs where nodes represent people and edges represent some type of connection between these people [1], such as friendship or common interests. Scientists in both academia and industry have recognized the importance of these networks and have focused on various aspects of social networks. One aspect that is often studied is the structure of these networks [1]–[12]. Previously, a *missing link problem* [1], [2] 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 node information. More recently a new *missing node identification* problem was introduced [13], [14] which locates and identifies missing nodes within the network. Previous studies have shown that combining the nodes' attributes can be effective when inferring missing links or attributes [7], [15]. We show how specific node attributes, such as demographic or historical information about specific nodes, can also be used to better solve the missing node problem, something that previous works did not consider.

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

Recent algorithms which were developed to solve similar problems, such as `MISC` [13] and `KronEM` [10], used the structure of the network but did not consider information about specific nodes. The `MISC` algorithm, which is most similar to our work, focused on a specific variation of the missing nodes problem where the missing nodes requiring identification are "friends" of known nodes. An unidentified 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. Following this approach, the missing node challenge is to try to determine which of the "placeholder" nodes are associated with the same unidentified friend. In other words, what is the correct clustering of the "placeholder" nodes? As was true in Eyal et al.'s work [13], we also assume that tools such as automated text analysis or image recognition software can be used to aid in generating placeholder nodes. For example, a known user makes reference to a coworker who is currently not a member of the network, or has friends which are not subscribers of Social News Inc. and are thus only visible as anonymous "users". Such mining tools can be employed on all of the nodes in the social network in order to obtain indications of the existence of a set of missing nodes.

The key contribution of this paper is how to integrate information about known nodes in order to help better solve the missing node problem. Towards this goal, we present three algorithms suitable for solving this missing node problem:

- *S.Sina and S. Kraus are with the Department of Computer Science, Bar Ilan University, Ramat-Gan, Israel 92500. E-mail: sinasi@macs.biu.ac.il, sarit@cs.biu.ac.il*
- *A. Rosenfeld is with the Department of Industrial Engineering, Machon Lev, Jerusalem, Israel, 91160. E-mail: rosenfa@jct.ac.il*

SAMI-A (**S**tructure and **A**ttributes **M**issing node **I**dentification using **A**ttributes' similarity), **SAMI-C** (**S**tructure and **A**ttributes **M**issing node **I**dentification using **C**oncatenate affinities), and **SAMI-N** (**S**tructure and **A**ttributes **M**issing node **I**dentification using social-attribute **N**etwork). The first algorithm, SAMI-A, calculates a weighted sum of two *affinity* components: one based on the network graph structure, as in previous work [13], and a new measure based on common attributes. The second algorithm, SAMI-C, concatenates the two *affinity* components: one based on the network graph structure and a new measure based on common attributes. The third algorithm, SAMI-N, combines the known nodes' attribute data into a Social-Attribute Network (SAN) – a data structure previously developed by [7], [16], [17]. We then again use a weighted sum of different components in the SAN to create the *affinity* measure.

We found that all clustering-based algorithms – all of the algorithms that we introduced, SAMI-A, SAMI-C and SAMI-N, as well as the MISC algorithm on which they were based – were each best suited for specific problem instances. Furthermore, we found that parameters in each of these algorithms might need tuning for different problem instances with different missing nodes or network sizes. Thus, an important question is to discover which of these algorithms, and which tuned parameter value in each algorithm, is best suited for a specific problem instance. Towards solving this problem, we present **OASCA**, an **O**nline **A**lgorithm **S**election for **C**lustering **A**lgorithms. While the idea of tuning an algorithm for a specific problem instance is not new, the application of these approaches to clustering algorithms is not trivial. During online execution, OASCA solves this challenge by using a novel relative metric to predict which clustering algorithm is best suited for a given problem instance. This facilitates effective selection of the best clustering algorithm.

2 Related Work

In solving the *Missing Node Identification* problem, this research is based on several existing areas of research for solving the challenge of identifying *Missing Information in Social Networks*. Specifically, we use variations of two existing research areas: clustering algorithms and metrics built for the missing link problem.

Many works have previously addressed the *Missing Information in Social Networks* problem, which attempts to uncover hidden information in social networks. One important aspect that is often studied is the structure of social networks [1]–[12]. Previously, a *Link Prediction* problem [1], [2] was defined as the attempt to locate the connections (edges) that 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. 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 according to which computation is best suited for predicting link similarity. For example, Liben-Nowell and Kleinberg [1] 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 the use of an adaptation of Adamic and Adar's measure of the similarity between webpages [18] and Katz's calculation for the shortest path information, which weighs the short paths more heavily [19] 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 [20] propose a method that performs well for detecting missing links as well as spurious links in complex networks. Their 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. [6] 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. [3] 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. [11] 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. [12] 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. [7] 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. [5] 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 [21] 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 we empirically showed that missing data causes errors in estimating these parameters. While advocating its importance, this work does not offer a definitive statistical solution

to overcome the problem of missing data.

In this work we present significant advancements in the state-of-the-art *Missing Node Identification* research. The missing node problem is relatively new, with only several works currently available on this subject [10], [13]. All of the works consider only the network structure in solving this problem. Eyal et al. [13], [14] presented the MISC algorithm, which was the first to develop how to use spectral clustering in the missing node identification problem. The spectral clustering algorithm of Jordan, Ng and Weiss [22] is a well documented and accepted algorithm, with applications in many fields including statistics, computer science, biology, social sciences and psychology [23]. The main idea behind Eyal et al.'s work [13], [14] was to embed a set of data points, which should be clustered, in a graph structure representing the *affinity* between each pair of points based on the structure of the network. One contribution of this paper is to consider how specific node attributes' data, such as demographic or historical information about specific nodes, can be used to better solve the missing node problem, something that Eyal et al. [13], [14] did not consider. Furthermore, in this work we demonstrate that using K-means clustering is more effective than spectral clustering, especially in solving the missing node problem in large-scale networks that need to consider the nodes' attributes data.

Kim and Leskovec [10] tackled the *network completion* problem, which is a similar problem that deals with situations where only a part of the network is observed and the unobserved part must be inferred. They proposed the KronEM algorithm, which uses an Expectation Maximization approach and where the observed portion of the network is used to fit a Kronecker graph model of the full network structure. The model is used to estimate the missing part of the network, and the model parameters are then re-estimated 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. Their research differs from ours in several key ways. First, and most technically, the KronEM prediction is based on link probabilities provided by the EM framework, while our algorithm is based on a clustering method and graph partitioning. Second, 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 networks with many missing nodes, SAMI is effective in local regions of the network with a small number of missing nodes where data mining can be employed. More importantly, as we previously found [24], our proposed algorithms, SAMI-A and SAMI-N (with the spectral clustering algorithm variation), can achieve significantly better prediction quality than KronEM or even the more closely related MISC algorithm [13], [14].

Recently, many studies have considered different ways to incorporate additional information in addition to the network structure in order to solve different problems related to the *Missing Information in Social Networks* problem. One area of research focused on the idea of using attributes of specific nodes. This idea was previously considered within different problems, however none of the studies considered using the information within the *Missing Node Identification* problem.

Several previous works [7], [16], [17] propose a model to jointly infer missing links and missing node attributes by representing the social network as an augmented graph where the nodes' attributes are represented as special nodes in the network. They show that link prediction accuracy can be improved when including the node attributes. In our work, we apply a similar approach in the SAMI-N algorithm, but infer the identity of missing nodes instead of missing links or missing node attributes. Other approaches studied different ways of leveraging information about known nodes within the network in order to better solve the missing link or missing attribute problems. For example, Freno et al. [5] 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. The proposed algorithm learns the similarity function for feature vectors using the visible graph structure. Backstrom and Leskovec [25] approach the predicting and recommending links problem. A link recommendation problem is a different way to view the missing link problem, where the aim is to suggest to each user a list of people with whom the user is likely to create new connections. They developed an algorithm based on supervised random walks that naturally combine the information from the network structure with both node and edge attributes.

Kim and Leskovec [15] developed a Latent Multi-group Membership Graph (LMMG) model with a rich node feature structure. In their model, each node belongs to multiple groups and each latent group models the occurrence of links as well as the node feature structure. They showed how LMMG can be used to summarize the network structure, to predict links between the nodes and to predict missing features of a node. Brand [26] proposed a model for collaborative recommendation. He studied various derived quantities and showed that normalized correlation-based rankings, such as angular-based quantity, are more predictive and robust to perturbations of the graph's edge set than rankings based on commute times, hitting times and related graph-based dissimilarity measures. Similarly, we also use a normalized measure in order to avoid biases towards nodes with high degrees, as can be seen in detail in Section 3.2.

A second key contribution of this paper is how to select, online and during task execution, the best clustering algorithm. We found that the previously developed MISC algorithm [13], as well as the SAMI-A, SAMI-C and SAMI-N extensions that we propose in this paper, are each best suited for specific clustering instances. Thus, a mechanism is needed to select the best algorithm for a given problem. Previously, Rice [27] generally defined the algorithm selection problem as the process of choosing the best algorithm for any given instance of a problem from a given set of potential algorithms. However, the key challenge is how to predict which algorithm will perform the best. Several previous works perform no prediction and instead run all algorithms for a short period in order to learn which one will be best for a given problem. For example, Minton et al. [28] suggested running all algorithms for a short period of time on the specific problem instance. Secondary performance characteristics were then compiled from this preliminary trial in order to select the best algorithm. Talman et al. [29] considered an agent that must choose which heuristic or strategy will help it the most in achieving its objectives. They proposed an algorithm for deciding how much information to acquire in order to make a decision while incurring minimal cost. Gomes and Selman [30] suggest running several algorithms (or randomized instances of the same algorithm) in parallel, thereby creating an algorithm portfolio. However, in our problem the true structure of the network is not known, making it impossible to predict which algorithm will definitively be best. Using algorithm selection in

conjunction with clustering algorithms has also recently begun to be considered. Halkidi and Vazirgiannis [31] considered how to determine the number of optimal clusters in a given clustering algorithm, such as K-means. Kadioglu et al. [32] considered how optimization problems could be solved through created clusters of optimal parameters. However, to the best of our knowledge, we are the first to consider how to select online between different clustering algorithms and between the parameters within each of these algorithms. This is the key contribution of the OASCA algorithm presented in this paper.

This work includes three significant differences compared to the preliminary results which were published in Sina et al. [24]: First, we were able to significantly improve the SAMI algorithm (section 4) in terms of runtime and memory consumption without performance degradation (see Table II). Second, we include extensive experiments with larger networks and larger numbers of missing nodes (section 8.3). Third, in this paper we also conduct a thorough study about dealing with partial and inaccurate information (section 9). Overall, this paper presents an extensive and thorough study for solving the missing node problem using structure and attribute information.

3 Overview and Definitions

In this section we define the missing node problem which we address. We also provide general formalizations about social networks and evaluation metrics used throughout the paper.

3.1 Problem Definition

We assume that there is a social network represented as an undirected graph $G = (V, E)$, in which $n = |V|$ and $e = \langle v, u \rangle \in E$ represents an interaction between $v \in V$ and $u \in V$. In addition to the network structure, each node $v_i \in V$ is associated with an attribute vector $\vec{AV}_i$ of length l. Referring back to the Social News Inc. example found within the Introduction, the social network contains nodes which are participants, and where edges are relationships and attributes are node specific information, such as a person's country of origin, skills and membership time in various professional groups in the network. We assume that each value in the attributes' vectors is binary, i.e. each node has or does not have a given attribute. Formally, we define a binary attributes matrix A of size nxl where $A_{i,j}$ indicates whether or not a node $v_i \in V$ has an attribute j. We choose to use a binary representation for the attributes in order to ease our implementation, as was done previously by other studies [7], [17]. Nevertheless, any other attribute type can be transformed into one or more binary attributes. We use discretization to transform all continuous real-value attributes, such as active time, into one or more binary attributes. For example, membership time can be quantified as binary values whether or not a person is a member within a specific group. Similarly, a person's membership time can be translated into three binary attributes – Long-time-Loyal-Customer, Medium-time-Customer and New-Member – using a threshold vector of size three. All categorical attributes, such as country, are transformed into a list of binary attributes, each for any value, e.g. USA, UK, Canada, where a given person does or does not live in that country.

Some of the nodes in the network are missing and are not known to the system. We previously defined this problem [13] by denoting the set of missing nodes as $V_m \subset V$, and we assume that the number of missing nodes is given[1] 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 $E_k = \{\langle v, u \rangle \mid v, u \in V_k \wedge \langle v, u \rangle \in E\}$. Towards identifying the missing nodes, we focus on a part of the network, $G_a = (V_a, E_a)$, that we define as being available for the identification of missing nodes. In this network, illustrated in Figure 1, each of the missing nodes is replaced by a set of placeholders. Formally, we define a set V_p for placeholders and a set E_p for the associated edges. For each edge $\langle v, u \rangle \in E$ where $v \in V_m$ is a missing node and $u \in V_k$, a placeholder is created. That is, for each original edge $\langle v, u \rangle$, we add a placeholder v' for v to V_p and connect the placeholder to the node u with a new edge $\langle v', u \rangle$, which we add to E_p. We denote the source of the placeholder, $v' \in V_p$, with $s(v')$. Putting all of these components together, $V_a = V_k \cup V_p$ and $E_a = E_k \cup E_p$. For a given missing node v, there may be many placeholders in V_p. The missing node challenge is to try to determine which of the placeholders should be clustered together and associated with the original v, thus allowing us to reconstruct the original social network G. To better understand this formalization, please again consider the Social News Inc. example from the Introduction. An edge between two users indicates that these two users communicated together. We might have additional information regarding the registered users, such as previous jobs, memberships, skills and non-professional interests. We consider the Social News Inc. subscribed users to be the known nodes and the other LinkedIn users are the anonymous users, for whom Social New Inc. only has references from its current subscribed users. We would like to identify which of the anonymous users are actually the same person. Thus, our purpose is to output a placeholder clustering C and a predicted graph $\hat{G} = (\hat{V}, \hat{E})$ where $\hat{V} = V_k \cup \{v_c | \text{a new node } v_c \text{ for each cluster } c \in C\}$ and $\hat{E} = E_k \cup \{(u, v_c) | \text{a new edge for each placeholder } v \in c, (u, v) \in E_v\}$.

3.2 Affinity Measures

Both the previously developed MISC algorithm as well as the SAMI-A, SAMI-C and SAMI-N algorithms proposed in this paper use *affinity* measures as part of their clustering algorithms. Specifically, these algorithms calculate an affinity measure between each pair of nodes in the network and send it to the clustering algorithm to determine which of the placeholders are associated with the same source node. Spectral clustering is a general algorithm used to cluster data samples using a certain predefined similarity (which is known as an *affinity* measure) between them. It creates clusters that maximize the similarity between points in each cluster and minimize the similarity between points in different clusters [22]. Thus, the success of the algorithm depends on the affinity matrix. While several affinity measures based on network structure have been studied previously [13], in this paper we use the two measures that have yielded the best results thus far: Relative Common Neighbors (RCN) [1] and Adamic/Adar (AA) [18]. Additionally, we use one affinity measure based on common attributes among the nodes (ATT). Note that this measure is not based on the general structure of the network, but on similarities between specific nodes' attributes.

These affinity measures are calculated between each pair of nodes, v_i and v_j, in the network. The Relative Common Neighbors measure, RCN_{ij}, calculates the number of common neighbors

1. Previous work [13] has found that this number can also be effectively estimated.

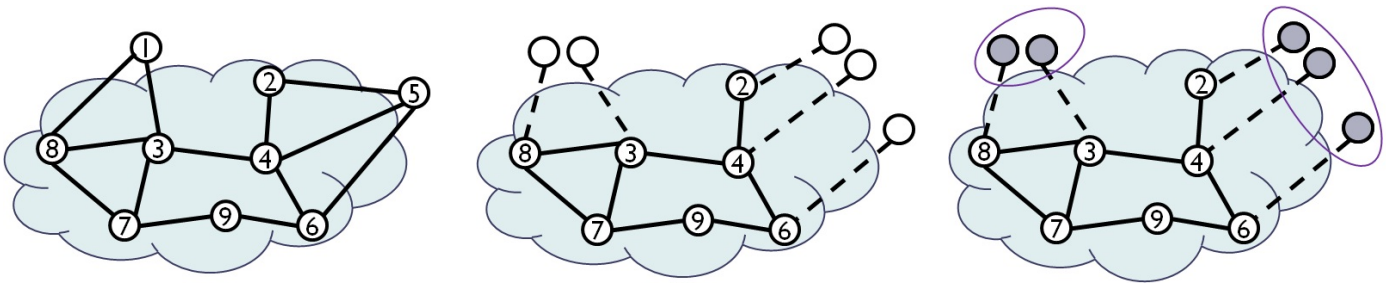

Fig. 1: A full network (on the left); the known network and the visible network obtained by adding the placeholders for the missing nodes 1 and 5 (in the middle); and the correct clustering of the placeholders (on the right). The placeholders in each cluster are combined into one node which represents a missing node.

between v_i and v_j. The Adamic/Adar measure, AA_{ij}, checks the overall connectivity of each common neighbor to other nodes in the graph and gives more weight to common neighbors who are less connected. The common attribute affinity measure, ATT_{ij}, is based on the nodes' attributes' similarity and it is defined as the number of common attributes between the two nodes divided by the size of the unified attribute set of the two nodes. This measure was inspired by the homophily relationship (love of the same) previously studied in [33]. Formally, let $\Gamma(i)$ denote the group of neighbors for a given node, v_i, in the network graph. We define the RCN_{ij}, AA_{ij} and ATT_{ij} affinity measures as: RCN_{ij}=$\frac{|\Gamma(i) \bigcap \Gamma(j)|}{min(|\Gamma(i)|,|\Gamma(j)|)}$, AA_{ij}=$\sum_{u \in \Gamma(i) \bigcap \Gamma(j)} \frac{1}{log(|\Gamma(u)|)}$ and ATT_{ij}=$\frac{|S(i) \bigcap S(j)|}{|S(i) \bigcup S(j)|}$. We consider the nodes that act as placeholders for missing nodes to be connected to their neighbor's neighbors for both the RCN and the AA measures, even though they only have one neighbor each. We divide the RCN measure by $min(|\Gamma(i)|, |\Gamma(j)|)$ to act as a normalizing effect in order to avoid biases towards nodes with a very large number of neighbors. In the ATT measure, $S(i)$ is defined as the set of attributes of node v_i. Note that we do not have attributes for nodes that are placeholders, thus to each placeholder we assign the attributes of its neighbor and optionally also the attributes of its neighbor's neighbors (depending on the algorithm preference). We will refer to these affinity measures matrices as M^{RCN}, M^{AA} and M^{ATT}, respectively.

3.3 Evaluation Measures

We considered two types of evaluation measures in order to gauge the effectiveness of the algorithms presented: *Graph Edit Distance* (GED) and *Purity*. GED compares the output graph of a given algorithm, $\hat{G} = (\hat{V}, \hat{E})$, to the original network graph, G, from which the missing nodes were removed. GED is defined as the minimal number of edit operations required to transform one graph to the other [34]. An edit operation is the 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 [34] to find an approximation of the GED. 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. It can be used to compare any two methods as long as they both produce a predicted graph. The disadvantage of computing the GED lies in its extended computational time. Due to this expense, a *purity* measure, which can be easily computed, can be used instead. *Purity* is an accepted measure of checking the quality and accuracy of a clustering-based algorithm [35]. The purity measure attempts to assess the quality of the predicted clustering (placeholders) compared to the true clustering (the missing nodes).

In evaluating our algorithms, we first consider the original network, then remove nodes to make them "missing". We can then evaluate how accurate our algorithms were in identifying the true structure of the network. Within the GED measure we check how many edit operations separate the two networks. The purity measure is calculated in two steps as follows: Step one – classify each cluster according to the true classification of the majority of samples in that cluster. Here, we classify each cluster according to the most frequent true original node $v \in V_m$ of the placeholder nodes in that cluster; Step two – 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 our problem setting, where c_k is defined as the set of placeholders which were assigned to cluster k, purity is defined as: $purity(C)$=$\frac{1}{|V_p|} \sum_k max_{v \in V_m} |c_k \cap \{v' \in V_p \mid s(v') = v\}|$.

4 THE SAMI ALGORITHMS

Algorithm 1 presents the pseudo code for the base of all three SAMI algorithm variations. The algorithm is based on the MISC algorithm [13] with two major changes: first, we add the attribute information of known nodes to the affinity matrix; and second, the SAMI algorithms no longer process the affinity measures of missing nodes by spectral clustering. Instead, they exclusively use the K-means clustering algorithm on the placeholder's affinity matrix. We chose to implement these changes for two reasons. First, we wanted to improve the performance of the original MISC algorithm when dealing with large scale networks. In MISC, spectral clustering was applied to the affinity matrix of the entire network– of both known nodes and placeholders. As we now apply K-means only on the matrix of a small number of placeholder nodes, we can efficiently process even larger networks. Second, we had to overcome a memory consumption challenge with the `SAMI-A` algorithm which was caused by the M^{ATT} matrix calculation. The analysis of the 2K training datasets shows that the attributes affinity matrix M^{ATT} is very dense (average of 60%-70%), while the structure affinity matrices M^{RCN} and M^{AA} are very sparse (average 4%-6%). Thus, the estimated memory for the attribute affinity matrix M^{ATT}, where n is the network size and d is the percentage of non-zero values (0.6-0.7), is $mem = n^2 * d * 8$ Bytes $\approx 5.5 * n^2$ Bytes. Accordingly, the estimated memory is $mem \approx$22MB for $n = 2K$, $mem \approx$1.4GB for $n = 16K$ and $mem \approx$55GB for $n = 100K$. In our previous work [24], we had two thresholds: *popularity* and *noise*, which

helped reduce the density of the attributes affinity matrix to 7%-15%. This maintained the advantage in the results as compared to the algorithm without attributes. However, this solution only enables us to scale up to 32K node networks.

Additionally, as we do not have attributes for nodes which are placeholders, we assume placeholders are similar to their neighbors and thus assign the placeholders the attributes of their neighbors. Furthermore, we considered assigning the attributes of the neighbors of link distance 2 – the placeholders' neighbor's neighbors. If these attributes conflicted, we assigned the placeholder both attribute values. In our preliminary tests, we found that overall SAMI−A and SAMI−N performed significantly better when assigning the placeholder the attributes of its neighbor, as opposed to assigning the attributes of its neighbor's neighbors. However, the SAMI−C variation overall performed significantly better when additionally assigning the attributes of its neighbor's neighbors. Thus, in our evaluation for the SAMI−A and SAMI−N algorithms, we assigned placeholders the attributes of the neighbor, but for the SAMI−C algorithm we also assigned the attributes of its neighbor's neighbors.

Algorithm 1 SAMI (**S**tructure and **A**ttributes **M**issing node **I**dentification)
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
V_p – the placeholder nodes
$\alpha : G(V, E) \longrightarrow \mathbb{R}^{|V_p| \times |V_p|}$ – a procedure for calculating the affinity matrix of placeholder 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: $\mathbf{A} \longleftarrow \alpha(G_a)$ – calculate the affinity matrix of the placeholder nodes in the graph
2: C $\longleftarrow$ *k_means(*$\mathbf{A}$*,N)* – cluster the rows that match the placeholder nodes to N clusters
3: $\hat{V} \longleftarrow V_k$, $\hat{E} \longleftarrow E_k$ – initialize the output graph to contain the known network
4: For each cluster $c \in C$ create a new node $v_c \in \hat{V}$
5: For each placeholder v in cluster c and edge $(u, v) \in E_a$, create an edge $(u, v_c) \in \hat{E}$
6: Return C, $\hat{G} = \left(\hat{V}, \hat{E}\right)$

We now present three approaches for adding node information: **SAMI−A**, **SAMI−C** and **SAMI−N**. The novelty of these algorithms lies in how they use the information to create new affinity measures to better solve the missing node problem.

4.1 The SAMI-A Algorithm

The first algorithm, **SAMI−A (Structure and Attributes Missing node Identification using Attributes' similarity)**, calculates an *affinity* measure based on a weighted sum of two components. The first component is based on the network structure, as in the MISC algorithm [13]. We implemented affinity measures based on RCN and AA (see Section 3 for definitions). The second component is based on the number of common attributes between two nodes. Formally, we define MA_{ij}^{RCN} and MA_{ij}^{AA} as: MA_{ij}^{RCN}=(1-w)RCN_{ij} + $wATT_{ij}$ and MA_{ij}^{AA}=(1-w)AA_{ij} + $wATT_{ij}$ where MA_{ij}^{RCN} and MA_{ij}^{AA} are the matrix of affinity measures for the SAMI−A algorithm using the RCN and AA measures, respectively. w is an input parameter which represents the relative weight of the attributes' similarity measure. It will determine how much weight will be given in the attribute matrix for the network structure (1-w) versus the attributes' information (w).

4.2 The SAMI-C Algorithm

The second algorithm, **SAMI−C (Structure and Attributes Missing node Identification using Concatenate affinities)**, creates the *affinity* measure as a concatenation of the structure and attributes affinity matrices instead of a weighted sum of the two components, as in SAMI−A. Formally, we define MC^{RCN} and MC^{AA} as: MC^{RCN}=$[M^{RCN} M^{ATT}]$ and MC^{AA}=$[M^{AA} M^{ATT}]$ where MC^{RCN} and MC^{AA} are the matrix of affinity measures for the SAMI−C algorithm using the RCN and AA measures, respectively.

4.3 The SAMI-N Algorithm

The third algorithm, **SAMI−N (Structure and Attributes Missing node Identification using social-attribute Network)**, combines the known nodes' attribute data into a Social-Attribute Network (SAN) – a data structure that was already developed [7], [16], [17]. We then use a uniform weighted sum of different components within the SAN to create the *affinity* measure. The algorithm first builds the SAN network from the original network and the attributes matrix. It starts with the original network G_v, where each original node and link in the SAN network are called a *social node* and *social link*, respectively. It defines a new *attribute node* for each binary attribute and adds it to the SAN network. It then adds a link – called an *attribute link* – between a social node and an attribute node if the social node has this attribute (i.e. TRUE value in the attributes matrix), as illustrated in Figure 2. As the SAN network

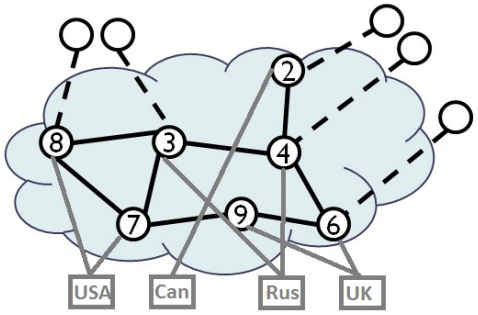

Fig. 2: Social-Attribute Network (SAN) with original attribute nodes.

has two types of nodes and links, social and attribute, it must adjust the affinity measures for this new type of network. This is done in line with previous work [7] with the option of giving weight to each node, whether social or attribute. Formally, we define the MN_{ij}^{RCN} and MN_{ij}^{AA} affinity measures as:

$$MN_{ij}^{RCN} = \frac{\sum_{u \in \Gamma(i) \bigcap \Gamma(j)} w(u)}{min(\sum_{u \in \Gamma(i)} w(u), \sum_{u \in \Gamma(j)} w(u))}$$

$$MN_{ij}^{AA} = \begin{cases} \sum_{u \in \Gamma(i) \bigcap \Gamma(j)} \frac{w(u)}{log(|\Gamma_s(u)|)} & \text{if } v_i, v_j \in V_v \\ \sum_{u \in \Gamma_s(i) \bigcap \Gamma_s(j)} \frac{w(u)}{log(|\Gamma(u)|)} & \text{else} \end{cases}$$

where MN_{ij}^{RCN} and MN_{ij}^{AA} are the matrix of affinity measures for the SAMI−N algorithm using the RCN and AA measures, respectively. $\Gamma(u)$ is defined as the group of neighbors of node u according to the SAN graph which includes both social and attribute links. $\Gamma_s(u)$ is defined as the group of social nodes which

are neighbors of node u according to the SAN graph, and $w(u)$ is node u's weight. Note that in our implementation, we use only one input parameter w, and we therefore use the same weight value, $w(u) = w/(1-w)$, for all of the attribute nodes and $w(u) = 1$ for all of the social nodes. We again divide by $min(\ldots)$ in order to avoid biases towards nodes with a very large number of neighbors.

5 THE OASCA ALGORITHM

In preliminary tests, we found that for different problem instances the best results are obtained from different clustering-based algorithms, including the SAMI-A, SAMI-C and SAMI-N algorithms we introduced, as well as the MISC algorithm on which they were based. We also found that the weight parameters within the SAMI-based algorithms might need to be tuned for different problems with varying numbers of missing nodes or network sizes. Thus, it is important to reveal which algorithm variation is best suited for a specific problem instance. Towards solving this problem, we present **OASCA**, an **O**nline **A**lgorithm **S**election for **C**lustering **A**lgorithms, which is based on the general algorithm selection approach previously proposed by Rice [27].

Following this approach, we define the OASCA algorithm as follows: First, OASCA runs the given portfolio of q clustering algorithms $\{CA_1 \ldots CA_q\}$ and saves the clustering results, C_i, of each algorithm. Specific to our missing node problem, C_i represents the output of the placeholders' clustering. In order to evaluate the algorithms' clustering results, we could not use the purity measure, since in a real world environment there is a lack of objective knowledge about the true original mapping of the placeholders. Thus, we had to define and calculate a novel measure, RS_i, which is based on a relative purity measure $RP_j(C_i)$ and forms the core of the OASCA algorithm. The $RP_j(C_i)$ measure assesses the quality of the clustering result C_i in relation to other portfolio algorithms' results. Formally, for each two clustering results, C_i and C_j, where $j \neq i$ and $s_j(v)$ is the source mapping of the placeholders according to the result C_j, we define: $RP_j(C_i) = \frac{1}{|V_p|} \sum_k max_{v \in V_m} |c_k \cap \{v' \in V_p \mid s_j(v') = v\}|$ and $RS_i = \sum_{j \neq i} RP_j(C_i)$. Lastly, OASCA returns the clustering results C^* with the highest score, i.e. $C^* = argmax_i RS_i$. Specifically, in this paper, we consider a portfolio which can include the MISC, SAMI-A, SAMI-C and SAMI-N algorithms, each with the two affinity types defined above (RCN and AA) and a set of weight values that are learned according to the procedure presented in Section 6.3.

6 EXPERIMENT METHODOLOGY

In this section, we describe the Steam social network used for evaluating the algorithms in this paper. We detail the methodology of our evaluation including the different networks considered and their missing nodes. As previously described, the SAMI-A and SAMI-N algorithms need to consider a weight, w, which will determine the weight that we will assign to the attributes' information in our algorithms. Specifically, in SAMI-A this value will determine the relative weight that will be given in the affinity matrix for the network structure (1-w) versus the attribute information (w), and in SAMI-N this weight determines the relative value of information in attribute nodes. Thus, in this section we also describe our procedure for learning the weight parameter values for the different algorithms and then we introduce MIK, an algorithm that exclusively uses K-means clustering to serve as a baseline similar to the MISC algorithm [13].

6.1 Dataset Description

We use a previously developed social network dataset, Steam [36] (http://steamcommunity.com), to empirically evaluate our work. The Steam community network is a large social network of players on the Steam gaming platform. The data we have is from 2011 and contains 9 million nodes ("anonymous" users) and 82 million friendship edges. Each user had the following data: country (the origin of the user; 50% voluntarily put country), member since (the date when the user opened his Steam account), a list of game playing times (number of hours played in the last two weeks) and a list of group memberships. We chose groups of attributes: country, playing time and player group association. These three groups form a total of 60 attributes – one for the country, another with 20 attributes of different game playing times and the third with 39 different official associations. As we are interested in studying the missing node problem where attribute information exists about known nodes, we had to ensure that the nodes within our dataset in fact contained such information. Towards this end, we crawled the social network and only selected nodes that have at least 2 games or groups. This crawling reduced the dataset size to 1.3 million nodes (users). The next challenge we had to address in using a dataset of this size was processing the data within a tractable period and overcoming memory constraints. To extract different networks' samples, we used a Forest Fire (FF) walk [37], [38], which starts from a random node in the dataset and begins 'burning' outgoing links and the corresponding nodes with a burning probability of 0.75. This is a variation of BFS walk, which randomly chooses only part of the node's outgoing links. We used this method as we want dense networks where each node has several links so that we can demonstrate the missing nodes problem, but still sample networks which preserve, as much as possible, the original dataset features. We crawled a 16K network from this reduced dataset, marked it as the *training* dataset and removed these nodes from the dataset. We then re-sampled this 16K node *training* dataset in order to extract several 2K *training* networks, which we used to learn parameters. Finally, we extracted several *test* networks with different sizes from the remaining dataset.

6.2 Experimental Setup

The experimental setup (see flowchart in Figure 3) starts by sampling a network from the dataset. This sampling is repeated ten times, each starting from a different random node, in order to avoid randomization errors and biases, creating ten different networks for each size. In the following step, N nodes are randomly selected as missing nodes. This is repeated ten times from each one of the networks, resulting in different random instances of the problem setting for each combination of graph size and N. The randomly selected missing nodes are removed from each instance. Each missing node is replaced with a placeholder for each of its links to remaining nodes in the network. The resulting graph is inputted into the 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, the SAMI algorithms create a predicted network. The goal of these algorithms is that the predicted network will resemble the structure of the original network from which the random nodes were removed. The clustering produced by the algorithms is evaluated using the purity measure described above, and the average purity value achieved for all of the instances

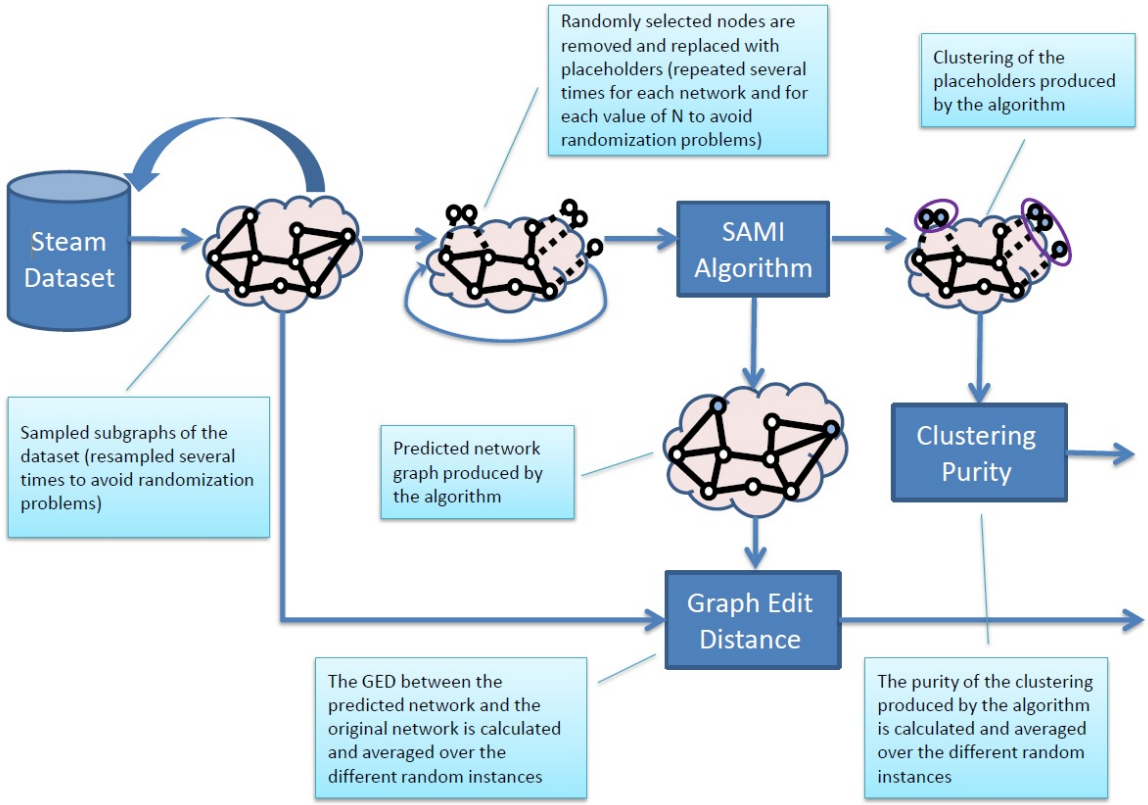

Fig. 3: A figure explaining the evaluation methodology for the experiments within this work.

and/or for the instances of a specific N is reported. The GED is also evaluated by comparing the predicted network to the original network. This result is also averaged over the different instances and/or for the instances of a specific N and is reported as well. In each experiment report, we indicate the network's size, the values of the N different missing nodes and the number of repetitions. Nearly all points in the tables and the graphs of the relevant figures below are the average of 100 runs (10 networks repeated 10 times) of the algorithm on randomly generated configurations.

6.3 Learning the Weight Parameter Values

We used the *training* datasets to empirically learn the best weight w for the SAMI-A and SAMI-N algorithms. Recall from Sections 4.1 and 4.3 that the weight w is used differently in the SAMI-A and SAMI-N algorithms. In the SAMI-A algorithm, the weight w is used for the weighted sum of the structure affinity and the attributes affinity. In the SAMI-N algorithm, we used the uniform weight $w(u) = w/(1-w)$ value for all attribute nodes and $w(u) = 1$ for all of the social nodes in the affinity measure calculation for the SAN network. We ran the algorithms with both affinity measure types, RCN and AA, using the 2,000 node training sample networks, where we randomly removed a set of missing nodes of sizes 10, 30, 50, 70, 100 and 150 which were, respectively, approximately 0.5%, 1.5%, 2.5%, 3.5% 5.0% and 7.5% of the network and a range of weights between 0.2 and 0.8 with 0.1 steps. Because we had only six training networks, we repeated the run 20 times. Table 1 shows the results for the 2,000 node training networks, where each value in the table represents the averages of all the runs for all of the missing node values (i.e. 6 missing node parameters X 6 training networks X 20 iterations per configuration = 720 runs). Based on these training results, we used the following weight parameters: w=0.4 for RCN SAMI-A, w=0.2 for RCN SAMI-N, w=0.8 for AA SAMI-A and w=0.2 for AA SAMI-N.

TABLE 1: The purity results for the 2,000 node training networks with different weights.

Weight	0.2	0.3	0.4	0.5	0.6	0.7	0.8
RCN SAMI-A	0.5806	0.5801	0.5824	0.5823	0.5816	0.5704	0.5316
RCN SAMI-N	0.5802	0.5801	0.5769	0.5716	0.5653	0.5558	0.5452
AA SAMI-A	0.5409	0.5428	0.5437	0.5443	0.5476	0.5468	0.5490
AA SAMI-N	0.5434	0.5424	0.5408	0.5395	0.5361	0.5318	0.5264

6.4 Comparison Configuration

We evaluated our four algorithms – SAMI-A, SAMI-C, SAMI-N and OASCA – using the known nodes' attribute information from the Steam dataset. For comparison, we first compared our SAMI algorithms to the original MISC algorithm and a variation of the SAMI-A and SAMI-N algorithms that are denoted SAMI-A-SC and SAMI-N-SC, respectively, which were based on the original MISC algorithm [14], [24]. However, because these algorithms are based on spectral clustering, they will not scale well to larger networks with the known nodes' attribute information due to the inherent high overhead with the full affinity matrices. Thus, we also evaluated a variation of the MISC algorithm, that we term MIK, which skips the spectral clustering step of the original algorithm and uses the K-means clustering algorithm directly on the placeholder structure affinity matrix. This enables a fairer comparison with the original algorithm yet facilitates a proper evaluation on larger networks. We chose not to compare our new algorithms to KronEM, another recently developed algorithm which only uses the network graph structure, and does not use attributes. This is

due to the fact that we already showed in our previous work [24] that both MISC and SAMI-A-SC outperform KronEM. As stated in that study, the KronEM algorithm accepts a visible graph and the number of missing nodes as its input. Consequently it is not based on the existence of placeholders and thus does not use them. Accordingly, it is not surprising that KronEM's performance does not improve when attributes from known nodes are considered. Finally, we also considered a Random assignment algorithm that assigns each placeholder to a random cluster. This algorithm is a baseline that represents the most naive of assignment algorithms. All algorithms based on affinity have two variations according to the network graph structure affinity matrix – Relative Common Neighbors measure (RCN) or Adamic/Adar (AA). The OASCA algorithm was evaluated with a portfolio that is based on variations of SAMI-A, SAMI-C, SAMI-N or MIK algorithms, each with RCN and/or AA measures.

7 Comparison Results

We first compared the K-means-based algorithms with the spectral-clustering-based algorithms. Specifically, we compared MISC, SAMI-A-SC, SAMI-N-SC, MIK, SAMI-A and SAMI-N with RCN and AA measures. We used the best weight from the training experiment for each of the two variations of SAMI-A and SAMI-N shown in Table 1. We ran 10 networks of 2,000 nodes where we randomly removed N missing nodes, using 5 missing node values of 10, 20, 30, 40 and 50, which respectively represented 0.5%, 1.0%, 1.5%, 2.0% and 2.5% of the network. We repeated each configuration 10 times to attain 100 results for each one of the missing node values. Table 2 shows the results (the higher the better) for the 2,000 node networks with RCN (above) and AA (below) measures. Each value in the table represents the average purity value over all of the runs (100 samples). We empirically observed that the RCN measure is successful even without MISC's spectral clustering preprocessing stage, while the AA measure is less successful and evidently requires this stage for its success. For the RCN measure, the K-means-based algorithms yielded better results than the spectral-clustering-based algorithms.

Overall, SAMI-A and SAMI-N performed significantly better than MIK (the ANOVA results for the mean difference of SAMI-A and SAMI-N compared to MIK at the 0.05 significance level were p=0.02 and 0.04, respectively). For the AA measure, SAMI-A and SAMI-N also performed significantly better than MIK (the ANOVA results for the mean difference of SAMI-A and SAMI-N compared to MIK at the 0.05 significance level were p=1.29E-15 and 5.13E-3, respectively). However, spectral-clustering-based algorithms performed better than the K-means-based algorithms, with SAMI-A-SC achieving the best results. Nonetheless, overall RCN SAMI-A and RCN SAMI-N performed significantly better than AA SAMI-A-SC (the ANOVA results for the mean difference at the 0.05 significance level were p=4.22E-8 and 1.13E-7, respectively).

We proceeded to compare the OASCA algorithm using ten samples of the 2,000 node networks with the same number of missing nodes. In these runs, we used the 3 weight values for each of the two variations of SAMI-A and SAMI-N, and we ran the OASCA algorithm to empirically confirm its effectiveness with a portfolio of q=14, which includes SAMI-A with 3 weight values, SAMI-N with 3 weight values and MIK algorithms with both the RCN and AA measures. We also ran the OASCA algorithm with the spectral-clustering-based algorithms, which we label OASCA-SC, with a portfolio of q=14, which includes the SAMI-A-SC with 3 weight values, SAMI-N-SC with 3 weight values and MISC algorithms. Figure 4 shows the purity results for the OASCA and OASCA-SC algorithms for each option of missing nodes and the overall average result. It is apparent that OASCA achieved better results (the higher the better) than OASCA-SC for all of the missing node values. These results were significantly better where the ANOVA results for the mean difference of OASCA from all of the runs and missing nodes' values compared to OASCA-SC at the 0.05 significance level was p=2.72E-7. Moreover, the OASCA results were also significantly better than the RCN SAMI-A and RCN SAMI-N algorithms (the ANOVA results for the mean difference at the 0.05 significance level were p=2.31E-2 and 4.48E-4, respectively). Note that the results of all of the algorithms with or without attributes were significantly better than the Random algorithm whose average purity results varied between 0.3921 for 10 missing nodes and 0.3128 for 50 missing nodes.

TABLE 2: Results of the comparison for the 2,000 node networks with RCN (above) and AA (below) measures.

RCN	Using Spectral Clustering			Using K-Means			
Missing Nodes	MISC	SAMI-A-SC	SAMI-N-SC	MIK	SAMI-A	SAMI-N	Random
10	0.6817	0.6978	0.6933	0.7276	0.7324	0.7307	0.3921
20	0.6409	0.6553	0.6512	0.6683	0.6705	0.6727	0.3430
30	0.6047	0.6202	0.6147	0.6337	0.6351	0.6350	0.3288
40	0.5854	0.5939	0.5869	0.6047	0.6068	0.6056	0.3160
50	0.5691	0.5754	0.5693	0.5831	0.5861	0.5849	0.3128
Average	**0.6164**	**0.6285**	**0.6231**	**0.6435**	**0.6462**	**0.6458**	**0.3385**

AA	Using Spectral Clustering			Using K-Means			
Missing Nodes	MISC	SAMI-A-SC	SAMI-N-SC	MIK	SAMI-A	SAMI-N	Random
10	0.6645	0.7006	0.6371	0.6570	0.6810	0.6688	0.3921
20	0.6191	0.6588	0.6336	0.6068	0.6183	0.6116	0.3430
30	0.5890	0.6284	0.6102	0.5772	0.5847	0.5746	0.3288
40	0.5683	0.6082	0.5898	0.5488	0.5615	0.5545	0.3160
50	0.5475	0.5873	0.5764	0.5342	0.5403	0.5344	0.3128
Average	**0.5977**	**0.6367**	**0.6094**	**0.5848**	**0.5972**	**0.5888**	**0.3385**

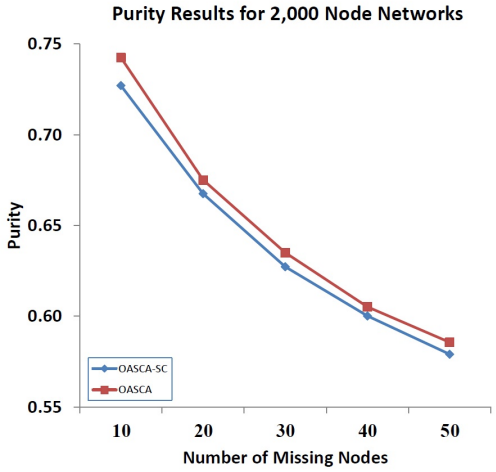

Fig. 4: Results of the comparison for the the 2,000 node networks with OASCA and OASCA-SC.

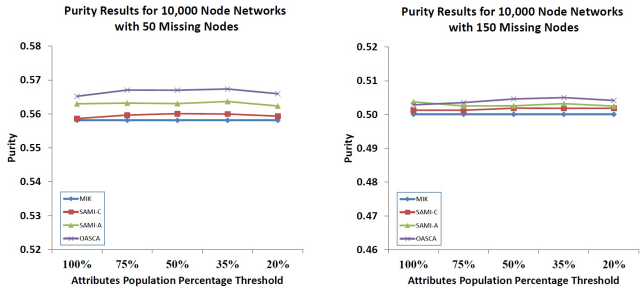

Fig. 5: Purity results for the 10,000 node networks with 50 (left) and 150 (right) missing nodes and with a different attribute population percentage threshold.

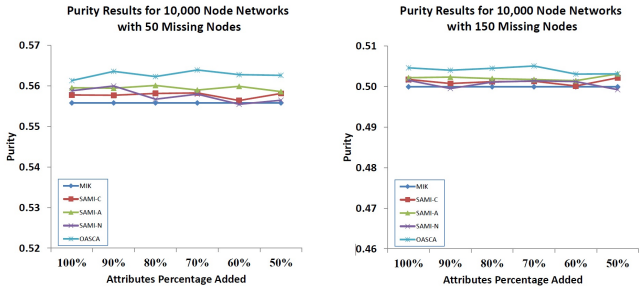

Fig. 6: Purity results for the 10,000 node networks with 50 (left) and 150 (right) missing nodes and with different percentages of assigned attributes.

8 Evaluation and Results

We then tested our SAMI-based algorithms and the OASCA algorithm with different configuration setups in order to evaluate the effect of the different parameters that might influence the algorithm's performance. Consequent to the results provided in section 7, which show that the RCN measure always outperforms the AA measure in the SAMI-based algorithms using K-means, the following experiments were performed with the RCN measure only.

8.1 Dealing with Frequent Attributes

We assessed whether attributes with high frequency can impact the algorithms' results. We presumed that including popular attributes in the affinity matrix would negatively affect the performance as they would add noise in the clustering algorithm, causing false indications that nodes containing the frequent information are similar. To evaluate this possibility, we considered a *popularity* threshold, which removes attributes from the matrix that appear with greater frequency than in the current network. We tested this hypothesis on 10,000 node networks with different population thresholds and randomly removed a set of missing nodes of sizes 10, 20, 30, 40, 50, 70, 100 and 150.

Figure 5 shows the average purity results (the higher the better) for MIK, SAMI−C, SAMI−A and OASCA algorithms with the RCN measure for each attributes' population percentage threshold option for the 50 missing nodes (left) and for the 150 missing nodes (right). The results show that when we removed the most popular attributes we could improve our results, and when the number of missing nodes increases, it is better to use a lower population percentage threshold, as the likelihood of false positives increases with more placeholder nodes. For example, when the number of missing nodes is greater than or equal to 70 or 100, the results for SAMI−A with a *popularity* threshold of 50% are significantly better than the results without a threshold (the ANOVA results for the mean difference of SAMI−A without a *popularity* threshold compared to a *popularity* threshold of 50% when the number of missing nodes $\geq$ 70 or $\geq$ 100 at the 0.05 significance level were p=4.06E-2 and 1.32E-2, respectively). And when the number of missing nodes is greater than or equal to 100, the results for OASCA with a *popularity* threshold of 50% and 35% are significantly better than the results without a threshold (the ANOVA results for the mean difference at the 0.05 significance level were p=3.12E-3 and 2.12E-2, respectively). These results again confirm that the OASCA algorithm performs best when followed by the SAMI−A algorithm.

Overall, the evaluation shows that high frequency attributes can negatively impact the average purity results and using a *popularity* threshold can improve the results. We conclude that when the number of missing nodes increases, it is better to use the less popular attributes. Thus, in the consequent experiments we used a *popularity* threshold of 50% for small to medium numbers of missing nodes, and a *popularity* threshold of 35% when a large number (greater than 150) of missing nodes needed to be identified.

8.2 Assigning Attributes to the Placeholder Nodes

As noted, we lack attribute information regarding the placeholder nodes and consequently we assign the attributes of neighbors to these nodes in SAMI−A and SAMI−N as well as the attributes of the node's neighbors' neighbors in SAMI−C. In these experiments we evaluated the impact of assigning attributes on the algorithms' results in order to determine whether all of the neighbors' attributes or only partial attributes (based on uniform distribution instead of popularity) should be assigned to the placeholders to improve the results. We used a percentage parameter, with values between 100% and 50%, as a random selection threshold for assigning the attributes to the placeholder nodes. We ran a test with the 10,000 node networks with a 50% *popularity* threshold and randomly removed the set of missing nodes of sizes 10, 20, 30, 50, 70, 100 and 150 as in the previous experiment.

Figure 6 depicts the average purity results for MIK, SAMI−C, SAMI−A, SAMI−N and OASCA algorithms with the RCN measure for each percentage threshold option for 10,000 node networks with 50 missing nodes (left) and 150 missing nodes (right). As expected, the results demonstrate that the percentage of attribute assignment has a higher impact on SAMI−A and SAMI−N algorithms, which only assign the placeholder the attributes of its neighbor, than on SAMI−C, which also assigns the placeholder the attributes of its neighbor's neighbors, and thus has information redundancy and lower sensitivity. Specifically, the results are significantly worse for SAMI−A and SAMI−N when the percentage value is 60% (the ANOVA results for the mean difference of SAMI−A and SAMI−N when using 100% of the attributes compared to only 60% of the attributes at the 0.05 significance level were p=2.77E-2 and 3.87E-2, respectively). It also shows that SAMI−N, which uses the attributes as embedded nodes in the SAN network, has a more complex dependency on the specific assigned attributes and its results are less stable than those of SAMI−A. For example, the results with 50 missing nodes (on the left) are lower with 80% assignment than with 70% assignment

and the results with 150 missing nodes (on the right) are lower with 90% assignment than with 80% assignment. For the SAMI−C algorithm, where we assigned attributes of the neighbors of a link distance of 2 – the placeholders' neighbor's neighbors, we obtained stable results when the percentage value was between 100% and 70% for all missing nodes. For a percentage of 60% the results were slightly worse, while for a percentage of 50% the results were slightly better. Nonetheless neither of the results are significantly different than the results of SAMI−C with 100% of the attributes. For the OASCA algorithm, when the number of missing nodes is 150 (on the right), the best results were obtained with 70%; however, this result is not significantly better compared to the results when all attributes were assigned. Additionally the results for OASCA were significantly worse when the percentage value was 60% (the ANOVA result for the mean difference at the 0.05 significance level was p=3.10E-2). We conclude that a percentage value of 70% for attribute assignment can improve OASCA's results without significant change in the results of the other algorithms. Also, since the results of SAMI−N were not better than the results of SAMI−A, i.e., they were less stable, and also took longer to calculate as a result of the need to reconstruct the network for each iteration, we decided not to run the SAMI−N algorithms in the subsequent experiments.

8.3 Scaling Up to More Missing Nodes and Large-Scale Networks

Subsequently, we evaluated the impact of the number of missing nodes and the network size on the algorithms' results. We presumed that as the number of missing nodes increases, the additional information gained from the algorithms from the known nodes' attributes decreases and may in fact become detrimental. The reasoning behind this supposition is that as the number of placeholders increases, the probability of finding nodes with similar attributes in other placeholders inside the network increases. Thus, in large scale networks the number of missing nodes rises in tandem with their placeholders, creating an increased likelihood of false positives. As this occurs, we postulate that the SAMI-based algorithms would at some threshold underperform in comparison to the MIK algorithm. While the MIK algorithm only considers the network structure, the SAMI algorithms additionally consider attributes. As the number of missing nodes increases, the large amounts of additional information considered by the SAMI algorithm will generate progressively more false positives because of the similarity of information in the placeholders. We explored this possibility by testing configurations with 25,000 and 100,000 node networks with a 35% *popularity* threshold and randomly removed a set of missing nodes of sizes 50, 100, 200, 300 and 500.

Figure 7 shows the average purity results for MIK, SAMI−C and SAMI−A for the 25,000 node (on the left) and 100,000 node (on the right) networks. For the 25,000 node network, the results of the OASCA algorithm were significantly better than the MIK and SAMI−C algorithms when the missing nodes were less than or equal to 300 (the ANOVA results for the mean difference of OASCA compared to MIK and SAMI−C at the 0.05 significance level were p=2.91E-20 and 2.82E-13, respectively). Nonetheless, for the 500 missing node problem, OASCA and MIK performed almost the same, and the results of SAMI−C were significantly worse (the ANOVA results for the mean difference of OASCA compared to SAMI−C at the 0.05 significance level was p=6.88E-27). However, the results of SAMI−A were equal to OASCA for

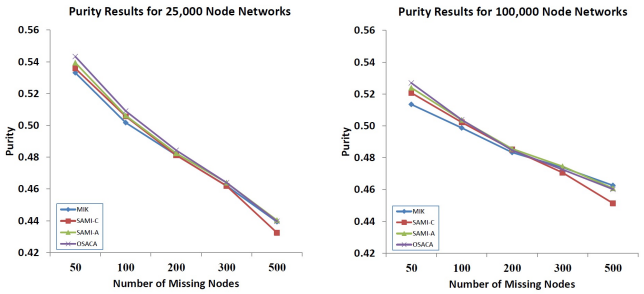

Fig. 7: Purity results for the 25,000 node (left) and 100,000 node (right) networks with missing nodes.

300 missing nodes and significantly better for 500 missing nodes (the ANOVA results for the mean difference of SAMI−A compared to OASCA and MIK at the 0.05 significance level were p=0.0078 and 0.0128, respectively). For the 100,000 node networks, the results for the OASCA algorithm were significantly better than the MIK, SAMI−A and SAMI−C algorithms when the missing nodes were less than or equal to 100 (the ANOVA results for the mean difference of OASCA compared to MIK, SAMI−A and SAMI−C at the 0.05 significance level were p=4.846E-16, 6.0E-3 and 9.12E-5, respectively). However, for the 200 and 300 missing nodes, the results of SAMI−A were significantly better (the ANOVA results for the mean difference of SAMI−A compared to OASCA, MIK and SAMI−C at the 0.05 significance level were p=1.27E-6, 2.7E-3 and 5.00E-5, respectively). Then again, the results for the 500 missing node configuration show that the MIK algorithm, which does not use the attribute data, performs significantly better than all of the other algorithms (the ANOVA results for the mean difference of MIK compared to OASCA, SAMI−A and SAMI−C at the 0.05 significance level were p=2.67E-6, 9.46E-5 and 1.35E-38, respectively). In light of these results we conclude that the use of the known nodes attribute data, which in our case is mainly group memberships and game play times, can improve the results for networks with no more than several hundred missing nodes. However, when the number of missing nodes increases beyond this point, especially in a large scale network, using attributes which are not anchored in the network structure can negatively affect the results.

9 Partial and Inaccurate Information

We also studied how partial and inaccurate information within the placeholders and the known nodes' attributes data can impact the algorithms' accuracy. With the algorithms and experiments described previously the assumption was that all placeholders' information can be identified with complete certainty and all of the known nodes' attributes data is available. Thus, when a node was removed from the original graph in the experiments described above, a placeholder was correctly connected to each one of the removed node's neighbors. Realistically, assuming 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 missing node information will be partial and noisy. We considered four different types of uncertainty regarding information within the network. In the first case, we considered the possibility that insufficient placeholders exist to correctly identify all missing nodes. We then considered a second case where not all of the known nodes' attributes data is available. Third, we considered

a case where extra placeholders exist. In this case, we assumed that the actual missing nodes are found in the set of placeholders, but extraneous information exists about additional placeholders which do not correspond to actual nodes in the network. Last, we considered a case where extraneous information exists about known nodes. For the first case (insufficient placeholders), we used evaluations based on Graph Edit Distance, as the actual number of placeholders is unknown based on the data. In this problem, the purity measure is not appropriate as it requires knowledge about the actual number of missing nodes and placeholders in its calculation. In contrast, the Graph Edit Distance measure 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. For the other uncertainty problem categories, e.g., where there are missing attributes, extra placeholders or attributes, we again assumed that the number of placeholders is known, allowing us once again to consider the purity measure in our evaluation.

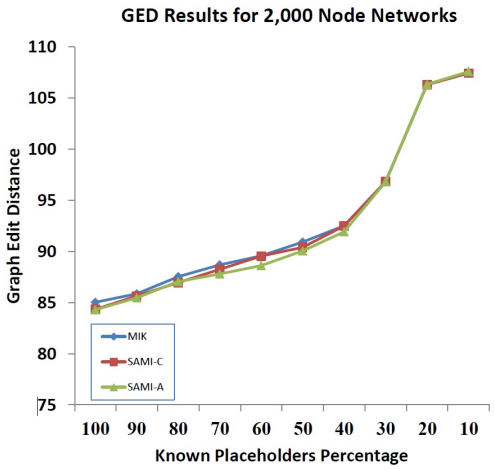

Fig. 8: GED results for the 2,000 node networks with a different known placeholder percentage.

9.1 Addressing Missing Placeholders

Under the assumption that a data mining module provides us with indications of the existence of placeholders, scenarios will likely occur whereby the module provides incomplete results, whereby some of the placeholders are not generated. 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 of wrong/noisy detection of placeholders. To assess how robust the SAMI-based algorithms are for 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 we ran 10 networks with a size of 2,000 nodes ten separate times, where we randomly removed a set of missing nodes of sizes 10, 20, 30, 40 and 50 and the percentage of known placeholders ranged from 10% to 100% of the total placeholders that should have been created when removing the missing nodes. Figure 8 displays the Graph Edit Distance (the lower the better) achieved by the MIK, SAMI-C and SAMI-A algorithms for each percentage of known placeholders. Each point in the graph represents the average Graph Edit Distance achieved for the 100 runs. The results show that SAMI-A achieved the best results (the lower the better) when the percentage of known placeholders ranged from 100% to 40%. However, the results 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 $\hat{G}$. Moreover, once 30% of the placeholders are unknown, the SAMI-based algorithms perform the same as the MIK algorithm, which does not use the known nodes' attribute data.

This scenario raises questions regarding the algorithms' ability to address missing information with large percentages of placeholders. As some placeholders are unknown, the resulting graph predicted by the algorithm would lack the edges between each unknown placeholder and its neighbor. To formulate this new problem we altered 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 defined 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 added 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 defined 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, otherwise it returns 0, i.e: $I(v) = \begin{cases} 1 & if\ \exists u \in V_p^u \ \wedge\ \langle v, u\rangle \in E_p^u \\ 0 & otherwise \end{cases}$

The value of $I(v)$ is unfortunately unknown to the system in this scenario. Instead, we modeled the data mining module's knowledge as $S(v) = I(v) + X(v)$, a noisy view of $I(v)$ with additive random noise $X(v)$, where X is an unknown random variable. The formal definition for this problem setting is thus: 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 $V_a \backslash V_k$ into 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 GED from the original network G.

In previous work [14], we proposed two possible compensation algorithms for solving this problem. The first algorithm, *Missing Link Completion*, added the additional step of missing link prediction to the existing missing node algorithm in order to complete the edges that were missing due to the unknown placeholders. The second algorithm, *Speculative MISC*, used a different approach for $S(v)$, where a new placeholder was added and connected to every node v whose indication value $S(v)$ was greater than T. Next, the original (MISC) algorithm was used on the new graph to predict the original graph. Our previous work showed that the *Speculative MISC* algorithm outperforms the *Missing Link Completion* by achieving a lower GED. Consequently in this work we chose to use the *Speculative MISC* algorithm as the basis for comparison. We extended this algorithm for the missing node with the attribute problem by using a new indicator $S^{ATT}(v)$ which also takes into account the attribute data. For each potential placeholder $v \in V_k$, we calculated its maximal attribute affinity with all of the known placeholders,

TABLE 3: GED results for the 2,000 node networks with 30% known placeholders and different compensation methods.

Average Graph Edit Distance Results	MIK	SAMI-C	SAMI-A
with full information	84.02	83.85	83.49
with 30% known PHs	96.27	96.28	96.32
with indicator based compensation	88.10	87.98	87.04
with indicator and attributes based compensation	---	87.47	**86.46**

$I^{ATT}(v) = argmax_u ATT_{vu}$ for all $u \in V_p^k$ and then we used the following integrated indicator $S^{ATT}(v)$ to add the potential missing placeholders, i.e. $S^{ATT}(v) = \frac{S(v)+I^{ATT}(v)}{2}$.

To study this point, we compare the MIK, SAMI-A and SAMI-C algorithms with full knowledge of the placeholders and where only 30% of the placeholders are known (the previous experiment showed a major deterioration in the results at 30%), with and without the two indicator-based compensation methods $S(v)$ and $S^{ATT}(v)$. We set the value of $X(v)$ with Gaussian noise with a mean of 0 and standard deviation of 1/4. Table 3 shows the results of average GED for all of the runs for all of the missing node values with 2,000 node networks. As expected, the best results (the lower the better) were achieved with full knowledge of the placeholders (first row). Again, it is apparent that when only 30% of the placeholders are known (second row), the three algorithms achieve almost the same results. The third row presents the results with the original compensation algorithm, which only uses the original indicator $S(v)$, and the fourth row depicts the results with the $S^{ATT}(v)$ indicator. The results with the $S^{ATT}(v)$ indicator are significantly better than the results with the original indicator $S(v)$ (the ANOVA results for the mean difference at the 0.05 significance level for SAMI-A and SAMI-C with the $S^{ATT}(v)$ indicator compared to the original indicator $S(v)$ were p=3.53E-2 and 1.16E-3, respectively). Overall when only 30% of the placeholders are known, the best results were achieved by the SAMI-C algorithm with the $S^{ATT}(v)$ indicator which were significantly better than the results of SAMI-C with $S^{ATT}(v)$ and MIK with $S(v)$ (the ANOVA results for the mean difference at the 0.05 level were p=1.17E-3 and 6.69E-7, respectively). The compensation algorithms enable us to mitigate the degradation in the results from over 15% down to 4%.

9.2 Addressing Missing Attributes

In the second scenario, as in the first, we again considered that not all of the known nodes' attribute data is available. As in Section 4, we assigned the placeholders the attributes of their neighbor in SAMI-C and also of their neighbor's neighbors in SAMI-A, thus considering two types of missing attributes – the attributes of the placeholder's neighbor and the attributes of the placeholders' neighbor's neighbors. To simulate this scenario, we used a uniform distribution of the placeholders' neighbors to remove a percentage of the nodes' attributes at either a distance of 1 or 2 from the placeholder representing the original true missing node. For example, if a given node X has one placeholder, in the distance 1 method we would remove a percentage of X's attributes (the placeholder's neighbor) and in the distance 2 method we would remove a percentage of the attributes of Y, which is a visible neighbor of X (placeholders' neighbors' neighbors). To study the impact of this factor we conducted a set of experiments where we ran 10 iterations of 10 networks with a size of 10,000 nodes and randomly removed N missing nodes, namely a set of missing nodes of sizes 10, 20, 30, 40, 50, 70, 100 and 150, and varied

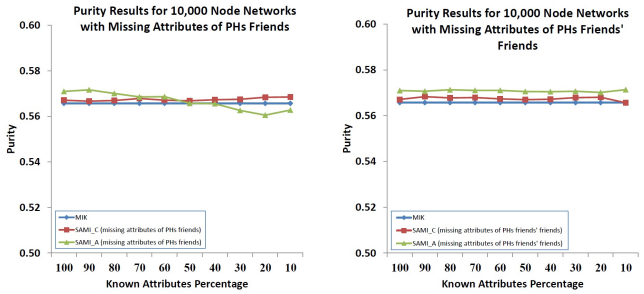

Fig. 9: Purity results for the 10,000 node networks with different known attribute percentages.

the known percentage of attributes from 100% to 10% (i.e. we removed from 0% to 90% of the attributes) for the two types of missing attributes.

Figure 9 shows the average purity results for the 10,000 node networks for each one of the assumed known attribute percentages for the two types of missing attribute information methods – on the left are the missing attributes of the placeholder's neighbor (distance 1), and on the right are the missing attributes of the placeholders' neighbors' neighbors (distance 2). The MIK results are shown as a baseline for the SAMI-based algorithms. As expected, the missing attribute information of distance 1 (on the left) has almost no effect on the results of SAMI-C, as this algorithm used the attributes of both distance 1 and distance 2, and thus used the existing attributes from distance 2 to remain effective. However, it is apparent that for SAMI-A, when the percentage of known attributes in distance 1 is 50% or less, the results degrade dramatically, and when the percentage of known attributes in distance 1 is 30% or less, the results are even worse than the results of MIK that does not use the attribute information. These results are compatible with the results of our presented tests in section 8.1 where we evaluated the popularity threshold and in section 8.2 where we evaluated how the assigned attributes can impact the algorithms' results. Here we removed the attributes with uniform distribution and not according to the popularity threshold as we did in Section 8.1. Thus though the degradation is apparent in the results in a smaller number of missing attributes, it is similar to the results presented in Section 8.1. For the missing attribute information of distance 2 (on the right), the results clearly demonstrate that the missing attributes had almost no impact on the results of both SAMI algorithms. There is a degradation in SAMI-C's results only when the known percentage of attributes in distance 2 is 10%. We conclude that the known node attributes can help improve the results compared to the MIK algorithm even when there is only partial data.

9.3 Evaluating the Effect of Extraneous Placeholders

In the third scenario, we considered the impact of having too many placeholders in the algorithms. In this possibility, we again assumed that some noise exists in the number of placeholders because of errors made by a data mining module in providing indications of possible missing nodes. However, this time we considered 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

TABLE 4: Purity results for the 10,000 node networks with different placeholder extraneous noise.

Average Purity Results	MIK	SAMI-C	SAMI-A
without extraneous placeholders	0.5729	0.5734	0.5771
5% global extraneous placeholders	0.5724	0.5740	0.5779
15% global extraneous placeholders	0.5740	0.5757	0.5770
25% global extraneous placeholders	0.5718	0.5745	0.5764
5% local extraneous placeholders (dist 2)	0.5683	0.5698	0.5720
15% local extraneous placeholders (dist 2)	0.5629	0.5636	0.5641
25% local extraneous placeholders (dist 2)	0.5575	0.5587	0.5585
5% local extraneous placeholders (dist 3)	0.5711	0.5723	0.5766
15% local extraneous placeholders (dist 3)	0.5684	0.5708	0.5723
25% local extraneous placeholders (dist 3)	0.5668	0.5668	0.5689

connected to the network nodes. The motivation behind the global method is that we wished to ascertain the impact of random noise in the network. Adding extraneous placeholders in this fashion replicates this type of noise. In the local methods we added extraneous placeholders in a uniform distribution to 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, in the distance 2 method we would add an extraneous placeholder to Y, which is a visible neighbor of X. In this experiment, once again we used 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, 40, 50, 70, 100 and 150. To 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 a distance of 2 and local with a distance of 3. We repeated this test 10 times. The results in Table 4 show the average mean purity for all of the runs. These results clearly demonstrate that the extraneous placeholders only had a slight effect on the algorithms' results. While the global extraneous placeholders had almost no impact, both local types of extraneous placeholders led to only a slight degradation in the results. These results contrast with the results in the previous section. Note that while having too few placeholders strongly degrades the results, having extraneous placeholders only slightly decreases the algorithms' performance.

9.4 Evaluating the Effect of Extraneous Attributes

In this last scenario, we again considered the impact of having extraneous attributes on the algorithms, for example due to noise in the known node attributes in a data mining module. However, this time we considered that false positives may provide incorrect values for attributes that in fact do not exist. To evaluate this possibility, similar to the previous case, we again considered three methods of how to add the extraneous attributes, one global method and two local methods. In the global method, we added extraneous attributes which were randomly connected to the network nodes. The motivation behind the global method is that we wished to ascertain the impact of random noise in the network. Thus, adding extraneous attributes in this fashion replicates this type of noise. In the local methods we added extraneous attributes in a uniform distribution to the nodes with existing placeholders. We added these extraneous attributes at either a distance of 1 (placeholder's neighbor) or 2 (placeholders' neighbor's neighbors) from the original true missing node (placeholder). For example, if a given node X has one placeholder, in the distance 1 method we would add an extraneous attribute to X and in the distance 2 method we would add an extraneous attribute to Y, which is a visible neighbor of X. In this experiment, we again used 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, 40, 50, 70, 100 and 150. To 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 a distance of 1 and local with a distance of 2. We repeated this test 10 times. The results in Table 5 show the average mean purity for all of the runs for the SAMI-A and SAMI-C algorithms. These results clearly demonstrate that the extraneous attributes had almost no impact on the algorithms' results. These results are compatible with our previous test, which show that the known nodes' attribute data, including redundancy, and removing part of the data or, in this case, adding some noise, only slightly affects the algorithms.

TABLE 5: Purity results for the 10,000 node networks with different attributes extraneous noise.

Average Purity Results	SAMI-C	SAMI-A
without extraneous attributes	0.5734	0.5771
5% global extraneous attributes	0.5735	0.5770
15% global extraneous attributes	0.5738	0.5754
25% global extraneous attributes	0.5733	0.5763
5% local extraneous attributes (dist 1)	0.5743	0.5762
15% local extraneous attributes (dist 1)	0.5736	0.5755
25% local extraneous attributes (dist 1)	0.5722	0.5748
5% local extraneous attributes (dist 2)	0.5735	0.5765
15% local extraneous attributes (dist 2)	0.5739	0.5761
25% local extraneous attributes (dist 2)	0.5736	0.5762

10 Conclusions and Future Work

We believe that this paper represents the first work that thoroughly examines the missing node identification problem by including information about both the network structure and attribute information of known nodes. The first key contribution of this paper is the method presented to integrate information about the known nodes in order to better solve the missing node problem. Towards this goal, we presented three clustering-based algorithms suitable for this problem – SAMI-A, SAMI-C and SAMI-N. All of these algorithms combine the nodes' specific attribute information in three ways. SAMI-A calculates a weighted sum of two *affinity* components, one based on the network graph structure and the other based on specific node attributes. SAMI-C creates the *affinity* measure as concatenation of the structure and attributes affinity matrices instead of using a weighted sum of the two components. SAMI-N first combines the known nodes' attribute data into a Social-Attribute Network (SAN) and then uses a weighted sum of different components in the SAN to create the *affinity* measure. We showed that SAMI-A, SAMI-C and SAMI-N outperform the recently introduced algorithms, MIK and MISC [13], which do not use the nodes' specific information.

The second key contribution of this paper is the novel OASCA algorithm provided, which is an online algorithm selection for clustering algorithms. Even though the RCN SAMI-A performed on average significantly better than other variations of SAMI-A, SAMI-C and SAMI-N algorithms or the MIK algorithm upon which it is based, each of the other algorithms were at times best suited for specific problem instances. We also found that parameters in each of these algorithms might need to be tuned for different problem instances with different missing nodes or network sizes. Thus, an important question was how to discover which of these algorithms, and which tuned parameter value in

each algorithm, is best suited for a specific problem instance. OASCA solved this challenge during online execution by using a novel relative measure to identify which clustering algorithm, from a given algorithm portfolio, is best suited for a given problem instance. OASCA allows us to choose, online, the best classifier without running all possibilities in advance. Though this was done in previous selection approaches [28], [30], [31] it was not applied a real-world environment.

Our evaluation shows that the OASCA algorithm gives the best results overall. Furthermore, the OASCA algorithm is a general online algorithm selection approach, and its input and output do not depend on the problem domain. Consequently, OASCA is also potentially suited for choosing from among other clustering algorithms in addition to the ones we present for the missing node identification problem. In the future, we would like to explore additional ways to improve the implementation of our proposed algorithms, SAMI-A, SAMI-C and SAMI-N, and continue evaluating these algorithms on additional datasets. We would also like to further explore the potential of the OASCA algorithm with a larger set of algorithms and additional datasets.

ACKNOWLEDGMENT

This research is based on work supported in part by MAFAT and the Israel Science Foundation grant #1488/14. Preliminary results of this research were published in the ASONAM 2013 paper entitled 'Solving the Missing Node Problem using Structure and Attribute Information'.

REFERENCES

[1] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American Society for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, May 2007.

[2] A. Clauset, C. Moore, and M. E. J. Newman, "Hierarchical structure and the prediction of missing links in networks," *Nature*, vol. 453, no. 7191, pp. 98–101, May 2008.

[3] M. Eslami, H. R. Rabiee, and M. Salehi, "Dne: A method for extracting cascaded diffusion networks from social networks," in *SocialCom/PASSAT*, 2011, pp. 41–48.

[4] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75 – 174, 2010.

[5] A. Freno, G. Garriga, C., and M. Keller, "Learning to Recommend Links using Graph Structure and Node Content," in *Neural Information Processing Systems Workshop on Choice Models and Preference Learning*, 2011.

[6] M. Gomez-Rodriguez, J. Leskovec, and A. Krause, "Inferring networks of diffusion and influence," *TKDD*, vol. 5, no. 4, p. 21, 2012.

[7] N. Z. Gong, A. Talwalkar, L. W. Mackey, L. Huang, E. C. R. Shin, E. Stefanov, E. Shi, and D. Song, "Predicting links and inferring attributes using a social-attribute network (san)," *CoRR*, 2011.

[8] V. Leroy, B. B. Cambazoglu, and F. Bonchi, "Cold start link prediction," *SIGKDD 2010*, 2010.

[9] M. A. Porter, J.-P. Onnela, and P. J. Mucha, "Communities in networks," *Notices of the American Mathematical Society*, vol. 56, no. 9, pp. 1082–1097, 2009.

[10] M. Kim and J. Leskovec, "The network completion problem: Inferring missing nodes and edges in networks," *SIAM International Conference on Data Mining (SDM), 2011*, 2011.

[11] E. Sadikov, M. Medina, J. Leskovec, and H. Garcia-Molina, "Correcting for missing data in information cascades," in *WSDM*, 2011, pp. 55–64.

[12] W. Lin, X. Kong, P. S. Yu, Q. Wu, Y. Jia, and C. Li, "Community detection in incomplete information networks," in *WWW*, 2012, pp. 341–350.

[13] R. Eyal, A. Rosenfeld, and S. Kraus, "Identifying missing node information in social networks," in *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[14] R. Eyal, A. Rosenfeld, S. Sina, and S. Kraus, "Predicting and identifying missing node information in social networks," *To Appear at ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2014.

[15] M. Kim and J. Leskovec, "Latent multi-group membership graph model," *arXiv preprint arXiv:1205.4546*, 2012.

[16] Z. Yin, M. Gupta, T. Weninger, and J. Han, "Linkrec: a unified framework for link recommendation with user attributes and graph structure," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 1211–1212.

[17] ——, "A unified framework for link recommendation using random walks," in *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conf. on*. IEEE, 2010, pp. 152–159.

[18] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social Networks*, vol. 25, no. 3, pp. 211–230, 2003.

[19] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, March 1953.

[20] R. Guimerà and M. Sales-Pardo, "Missing and spurious interactions and the reconstruction of complex networks," *Proceedings of the National Academy of Sciences*, vol. 106, no. 52, pp. 22 073–22 078, 2009.

[21] G. Kossinets, "Effects of missing data in social networks," *Social Networks*, vol. 28, pp. 247–268, 2003.

[22] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems 14*. MIT Press, 2001, pp. 849–856.

[23] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, December 2007.

[24] S. Sina, A. Rosenfeld, and S. Kraus, "Solving the missing node problem using structure and attribute information," in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ACM, 2013, pp. 744–751.

[25] L. Backstrom and J. Leskovec, "Supervised random walks: predicting and recommending links in social networks," in *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 2011, pp. 635–644.

[26] M. Brand, "A random walks perspective on maximizing satisfaction and profit," in *SIAM International Conference on Data Mining*, 2005, pp. 12–19.

[27] J. R. Rice, "The algorithm selection problem," in *Advances in Computers*, vol. 15, 1976, pp. 118–165.

[28] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird, "Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems," *Artificial Intelligence*, vol. 58, no. 1-3, pp. 161–205, 1992.

[29] S. Talman, R. Toister, and S. Kraus, "Choosing between heuristics and strategies: an enhanced model for decision-making," in *International Joint Conference on Artificial Intelligence*, vol. 19, 2005, pp. 324–330.

[30] C. P. Gomes and B. Selman, "Algorithm portfolios," *Artificial Intelligence (AIJ)*, vol. 126, no. 1-2, pp. 43–62, 2001.

[31] M. Halkidi and M. Vazirgiannis, "A data set oriented approach for clustering algorithm selection," in *PKDD*, 2001, pp. 165–179.

[32] S. Kadioglu, Y. Malitsky, M. Sellmann, and K. Tierney, "Isac - instance-specific algorithm configuration," in *ECAI*, 2010, pp. 751–756.

[33] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, pp. 415–444, 2001.

[34] O. Kostakis, J. Kinable, H. Mahmoudi, and K. Mustonen, "Improved call graph comparison using simulated annealing," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, ser. SAC '11, 2011, pp. 1516–1523.

[35] A. Strehl and J. Ghosh, "Cluster ensembles — a knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, pp. 583–617, Mar. 2003.

[36] R. Becker, Y. Chernihov, Y. Shavitt, and N. Zilberman, "An analysis of the steam community network evolution," in *Electrical & Electronics Engineers in Israel (IEEEI), 2012 IEEE 27th Convention of*. IEEE, 2012, pp. 1–5.

[37] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 177–187.

[38] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 631–636.