Solving the Missing Node Problem using Structure and Attribute Information

Sigal Sina
Bar-Ilan University
Ramat-Gan, Israel 92500
Email: sinasi@macs.biu.ac.il

Avi Rosenfeld
Jerusalem College of Technology,
Jerusalem 91160, Israel
Email: rosenfa@jct.ac.il

Sarit Kraus
Bar-Ilan University
Ramat-Gan, Israel 92500
Email: sarit@cs.biu.ac.il

***Abstract*—An important area of social networks research is identifying missing information which is not explicitly represented in the network, or is not visible to all. 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 could be useful in solving this problem. In this paper, we present two algorithms: `SAMI-A` and `SAMI-N`. Both of these algorithms use the known nodes' specific information, such as demographic information and the nodes' historical behavior in the network. We found that both `SAMI-A` and `SAMI-N` perform significantly better than other 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.**

I. 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], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [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. Most recently a new *missing node identification* problem [13] was introduced, 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], [14]. 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 work 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 nodes 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, `MISC` [13] and `KronEM` [10], which were developed to solve similar problems, used the structure of the network, but did not consider information about specific nodes. Our work extends Eyal et al.'s `MISC` algorithm [13]. The `MISC` algorithm 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 same unidentified friend. In other words, what is the correct clustering of the "placeholder" nodes? As was true in Eyal et al.'s work, 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 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 two algorithms suitable for solving this missing node problem: **`SAMI-A`** (**S**tructure and **A**ttributes **M**issing node **I**dentification using **A**ttributes' similarity) 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 between two *affinity* components, one based on the network graph structure, as was the case in previous work [13], and a new measure based on common attributes within known

nodes. The second algorithm, SAMI−N, combines the known nodes' attributes data into a Social-Attribute Network (SAN) – a data structure that was previously developed [7], [15], [16]. We then once again use a weighted sum between different components within the SAN to create the *affinity* measure.

However, we found that all clustering-based algorithms – both of the algorithms we introduced, SAMI−A 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 within 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 within 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.

II. Related Work

In solving the *Missing Node Identification* problem, we use variations of two existing research areas: spectral clustering algorithms and metrics built for the missing link problem. The spectral clustering algorithm of Jordan, Ng and Weiss [17] is a well documented and accepted algorithm, with applications in many fields including statistics, computer science, biology, social sciences and psychology [18]. Eyal et al. [13] presented the MISC algorithm, which was the first to develop how to use spectral clustering in the missing node identification problem. The main idea behind their work 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] did not consider.

Kim and Leskovec [10] tackled a similar problem, what they termed the *network completion* problem, which deals with situations where only a part of the network is observed and the unobserved part must be inferred. The proposed algorithm, called KronEM, uses an Expectation Maximization approach, where the observed part is used to fit a Kronecker graph model of the network structure. The model is used to estimate the missing part of the network, and the model parameters are then 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, our algorithm may be effective in local regions of the network with a small number of missing nodes where data mining can be employed. More importantly, and as our results detail, our proposed algorithms, SAMI−A and SAMI−N, can achieve significantly better prediction quality than KronEM or even the more closely related MISC algorithm. Similarly, our proposed algorithm offers a much lower time complexity, especially in comparison to the KronEM algorithm. While KronEM requires a running time of about 100-200 minutes in order to analyze a network of a few thousand nodes in our experiments, we were able to analyze such networks in less than 20 seconds on the same hardware. This low time complexity is achieved mainly by using a dimension reduction technique, which effectively causes the algorithm to focus on the local vicinity of the missing nodes. Such a technique cannot be easily combined into the KronEM method since it relies on the entire graph to calculate the Kronecker model [19].

The idea of using attributes of specific nodes was previously considered within different problems. Several previous works [7], [15], [16] 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 instead 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 over feature vectors using the visible graph structure. Kim and Leskovec [14] developed a Latent Multi-group Membership Graph (LMMG) model with a rich node feature structure. In this 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 the LMMG can be used to summarize the network structure, to predict links between the nodes and to predict missing features of a node. Another work, Brand [20] proposed a model for collaborative recommendation. He studied various quantities derived from the commute time and showed that angular-based quantity outperforms the commute time which is quite sensible to the node degree. In our case, in order to avoid biases towards nodes with high degree, we also use normilized measures.

A second key contribution of this paper is how to select, both online and during task execution, the best clustering algorithm. We found that the previously developed MISC algorithm [13], as well as the SAMI−A and SAMI−N extensions that we propose in this paper, are best suited for specific clustering instances, thus a mechanism is needed to select the best algorithm for a given problem. Previously, Rice [21] 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. [22] suggested running all algorithms for a short period of time on the specific problem instance. Secondary performance characteristics are then compiled from this preliminary trial in order to select the best algorithm. 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 [23] considered how to determine the number of optimal clusters within a given clustering algorithm, such as K-means. Kadioglu et al. [24] 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 within the `OASCA` algorithm presented in this paper.

III. 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.

`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. For example, in a social games network, nodes are players, edges are friendship relationships and attributes are node specific information, such as a player's country of origin, group membership and game playtime. We assume that each attribute in the attributes vectors is a binary attribute, i.e. each node has or does not have the 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. Nevertheless, any other attribute type can be transformed into one or more binary attributes. We use discretization to take all continuous real-value attributes, such as playtime, and transport them into one or more binary attributes. For example, game playtime can be translated into one binary variable, using a zero value threshold, where a user either plays or does not play this specific game, or it can be translated into three binary attributes – HeavyPlayer, ModeratePlayer and LitePlayer – using a threshold vector of size three. All categorical attributes, such as country, are transformed into a list of binary attributes, each for any origin value, e.g. USA, UK, Canada, where a given player did or did not originate from that country.

Some of the nodes in the network are missing and are not known to the system. We denote the set of missing nodes as $V_m \subset V$, and 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 the visible part of the network, $G_v = (V_v, E_v)$, that is known. In this network, 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 missing node $v \in V_m$ and for each edge $\langle v, u \rangle \in E$, $u \in V_k$, a placeholder is created. That is, 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 $s(v')$. When these components are considered together, $V_v = V_k \cup V_p$ and $E_v = E_k \cup E_p$. As 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

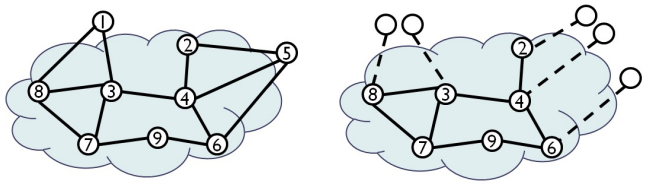

Fig. 1. A full network, the known network and the visible network obtained by adding the placeholders for the missing nodes 1 and 5.

formalization, please consider the following example: Assume we have a gamers network where users may register or not before playing a game. An edge between two users indicates that these two users play a game together. We might have additional information regrading the registered users, such as origin, group membership and playing time. We consider the registered users to be the known nodes, while the anonymous users are the placeholders. 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 $\hat{E} = E_k \cup \{(u, v_c) | \text{a new edge for each placeholder } v \in c, (u, v) \in E_v\}$.

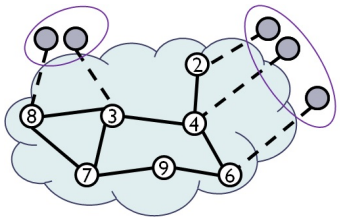

Fig. 2. Correct clustering of the placeholders. The placeholders in each cluster are united to one node which represents a missing node.

`Affinity Measures:` Both the previously developed `MISC` algorithm as well as the `SAMI-A` 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 pass it to the spectral 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

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

creates clusters which maximize the similarity between points in each cluster and minimize the similarity between points in different clusters [17]. Thus, the success of the algorithm depends on the affinity matrix. It is important to note that while the spectral clustering algorithm's goal is to cluster the placeholders, it uses information from all nodes, both the known ones and the placeholder ones, during the clustering process. While several affinity measures based on network structure were previously studied [13], in this paper we use the two measures that previously yielded the best results: the Relative Common Neighbors (RCN) measure [1] and the Adamic/Adar (AA) measure [25]. Additionally, we use one affinity measure based on common attributes between nodes (Att). Note that this measure is not based on the general structure of the network, but similarities between specific nodes' attributes.

These affinity measures are calculated between each pair of nodes, v_i and v_j, within the network. The Relative Common Neighbors measure, RCN_{ij}, calculates the number of common neighbors 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 attributes set of the two nodes. This measure was inspired by the homophily relationship (love of the same) previously studied [26]. 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:

$$RCN_{ij} = \frac{|\Gamma(i) \bigcap \Gamma(j)|}{min(|\Gamma(i)|,|\Gamma(j)|)}$$

$$AA_{ij} = \sum_{u \in \Gamma(i) \bigcap \Gamma(j)} \frac{1}{log(|\Gamma(u)|)}$$

$$Att_{ij} = \begin{cases} \frac{|S(i) \bigcap S(j)|}{|S(i) \bigcup S(j)|} & \text{if } v_i, v_j \in V_k \\ 0 & \text{else} \end{cases}$$

Since the nodes that act as placeholders for missing nodes only have one neighbor each, we also consider them to be connected to their neighbor's neighbors, for both the RCN and the AA measures. 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 which are placeholders. As a result, in the Att measure, if either v_i or v_j is a placeholder, then we assume this measure is 0.

`Evaluation Measures:` We considered two types of evaluation measures in order to measure 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 [27]. An edit operation is an addition or a deletion of a node or an edge. Since finding the optimal edit distance is NP-Hard, we use a previously developed simulated annealing method [27] 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 [28]. 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: $purity(C) = \frac{1}{|V_p|} \sum_k max_{v \in V_m} |c_k \cap \{v' \in V_p \mid s(v') = v\}|$.

IV. The SAMI Algorithms

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

A. 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 between 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 III 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:

$$MA_{ij}^{RCN} = \begin{cases} (1-w)RCN_{ij} + w * Att_{ij} & \text{if } v_i, v_j \in V_k \\ RCN_{ij} & \text{else} \end{cases}$$

$$MA_{ij}^{AA} = \begin{cases} (1-w)AA_{ij} + w * Att_{ij} & \text{if } v_i, v_j \in V_k \\ AA_{ij} & \text{else} \end{cases}$$

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. Because we do not have attributes for nodes which are placeholders, if one of the nodes, v_i or v_j, is a placeholder, we only use the affinity component which is based on the network structure.

B. The SAMI-N Algorithm

The second algorithm, **`SAMI-N`** **(Structure and Attributes Missing node Identification using social-attribute Network)**, combines the known nodes' attributes'

data into a Social-Attribute Network (SAN) – a data structure that was previously developed [7], [15], [16]. We then use a weighted sum between 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 *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 the

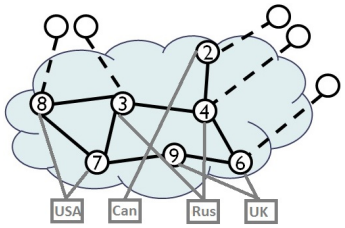

Fig. 3. Social-Attribute Network (SAN) with origin attribute nodes.

SAN network has two types of nodes and links, social and attributes, 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, 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, therefore we use the same weight value, $w(u) = w/(1-w)$, for all of the attributes 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.

V. The OASCA Algorithm

Based on preliminary tests, we found that the clustering-based algorithms, including the SAMI-A and SAMI-N algorithms we introduced as well as the MISC algorithm on which they were based, were each best suited for specific problem instances. Furthermore, we found that parameters within each of these algorithms might need to be tuned for different problems with differing numbers of missing nodes or network sizes. Thus, an important question is to discover which of these algorithms, and which tuned parameter value within 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, which is based on the general algorithm selection approach previously proposed by Rice [21].

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 keeps the clustering results, C_i, of each algorithm. Specifically, in our case the clustering results, C_i, represent the output of the placeholders' clustering. In order to evaluate the algorithms' clustering results, we could not use the purity measure, as in a real world environment we do not know 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, C_j where $j \neq i$ and $s_j(v)$ is the source mapping of the placeholders according to the result C_j: $RP_j(C_i) = \frac{1}{|V_p|}\sum_k max_{v\in V_m}|c_k \cap \{v' \in V_p \mid s_j(v') = v\}|$ and $RS_i = \sum_{j\neq i} RP_j(C_i)$. Last, 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 and SAMI-N algorithms, each with the two affinity types defined above (RCN and AA).

VI. Experiments Methodology

We use a previously developed social network dataset, Steam [29] (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), list of game playing times (number of hours played in the last two weeks) and list of group memberships. We choose groups of attributes: country, playing time and player group association. These three groups form a total of 60 attributes – one for the country, 20 attributes of different game playing times and 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 use a Forest Fire (FF) walk [30], [31], 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 use 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 crawl a 16K network from this reduced dataset, mark it as the *training* dataset and remove these nodes from the dataset. We then re-sample this 16K node *training* dataset to extract several 2K *training* networks, which we use for parameters learning. Last, we extract several *test* networks with different sizes from the remaining dataset.

A. Learning the Parameters' Values

We used the training datasets to empirically learn the domain dependent variables. These parameters include the optimal weight w for the SAMI−A and SAMI−N algorithms and two thresholds in order to optimize the memory consumption of SAMI−A's attribute affinity measure.

The weight parameter. It is important to point out 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 between the structure affinity and the attributes affinity. In the SAMI−N algorithm, we use the same 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 run these algorithms with both affinity measure types RCN and AA using the 2K training sample networks, the same 5 missing nodes values (11, 21, 31, 41 and 50) and a range of weights between 0.2 and 0.8 with 0.1 step. We repeat the run 5 times over the six training networks we had. Table I shows the results for the 2K training networks, where each value in the table represents the average over all of the runs for all of the missing node values (i.e. 150 runs). Based on this training data, we use w=0.8 for both AA SAMI−A and AA SAMI−N, w=0.3 for RCN SAMI−A and w=0.2 for RCN SAMI−N.

TABLE I. THE PURITY RESULTS FOR THE 2K TRAINING NETWORKS WITH DIFFERENT WEIGHTS

Weight	0.2	0.3	0.4	0.5	0.6	0.7	0.8
AA SAMI-A	0.6325	0.6305	0.6338	0.6351	0.6369	0.6357	0.6372
AA SAMI-N	0.5945	0.6000	0.5999	0.6072	0.6118	0.6150	0.6199
RCN SAMI-A	0.6283	0.6337	0.6309	0.6296	0.6296	0.6260	0.6259
RCN SAMI-N	0.6215	0.6178	0.6121	0.6078	0.6037	0.5993	0.5947

Thresholds for M^{Att}. We use the training datasets to optimize SAMI−A memory consumption. 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 attributes 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 for $n = 2K$ is $mem \approx$22MB, for $n = 16K$ is $mem \approx$1.4GB and for $n = 100K$ is $mem \approx$55GB. That means that before we could apply the SAMI−A algorithm for larger networks, we had to develop a method to reduce the density of the attribute affinity matrix. Using a combination of two techniques, which were tuned on the 2K training datasets, we succeeded in reaching a feasible density while keeping enough information for the algorithm to use. The first technique uses a *popularity* threshold. It reduces the number of nodes' attributes based on the attributes frequency in the current network. Our intuition for this technique was that attributes with high frequency contribute less information, thus we decided to filter out the attributes with a higher probability than an input percentage threshold. The evaluation showed that a *popularity* threshold of 20% reduced the density from 60%-70% to 20%-25%. The second technique was based on a fixed *noise* threshold value. Our intuition was that low attribute similarity measures may introduce noise to the affinity measures, thus we decided to filter out the entries in the attributes affinity matrix which were lower than a noise threshold. The evaluation showed that a *noise* threshold of 0.15 reduced the density from 20%-25% to 7%-15% and also improved the results as compared to a zero noise threshold. We use these two thresholds during the evaluation of larger networks described in section VII-B.

B. The Comparison Configuration

We evaluated our three algorithms – SAMI−A, SAMI−N and OASCA – which use the additional nodes' attributes information using the Steam dataset. For comparison, we also evaluated two recently developed algorithms – MISC and KronEM – using the same dataset, which only use the network graph structure and do not use attributes. We also considered a third Random assignment algorithm that assigns each placeholder to a random cluster. This algorithm is a baseline that represents the most naive of assignment algorithms. Note that the SAMI−A, SAMI−N and MISC algorithms each 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 which is based on variations of SAMI−A, SAMI−N or MISC algorithms, each with RCN and AA measures. It is also important to note that the KronEM algorithm accepts a visible graph and the number of missing nodes. This algorithm is not based on the existence of placeholders and therefore it does not use them[2]. The KronEM algorithm outputs a graph which predicts the missing nodes and their links. The KronEM algorithm also assumes that the number of nodes in the full graph is a power of two. Nonetheless, to facilitate a fair comparison between the different algorithms, we generated 10 networks, each containing $2^{11} = 2048$ $(2K)$ nodes sampled from the test dataset. We randomly removed N missing nodes from each network. The selected values for N were 11, 21, 31, 41 and 50 which are, respectively, approximately 0.5%, 1%, 1.5%, 2% and 2.5% of the network. The experiment was repeated five times for each selected value of N in each network, resulting in 50 iterations for each missing node percentage. Each resulting network was tested with all of the algorithms. The output of each algorithm was compared to the original network graph using Graph Edit Distance (GED) and Purity. Note that the purity measure is not applicable for the KronEM algorithms as it expected a placeholders' clustering, in addition to the predicted graph, and thus this measure was not calculated for the KronEM algorithm. While the MISC, SAMI−A, SAMI−N, OASCA and Random algorithms only predict the existence of links between the predicted nodes and the neighbors of the original missing nodes, the KronEM algorithm, on the other hand, might predict an erroneous link between a predicted node and any other node in the network which was not connected to the missing node. This is due to the fact that KronEM does not take advantage of the placeholders. Therefore, in order to fairly compare the KronEM with the other algorithms when calculating the GED, we only considered edit operations relating to links between the predicted nodes and the neighbors of the missing nodes. In addition, even though KronEM predicts a directed graph, we treated each predicted link as undirected, since this slightly improved the results of KronEM.

[2]In running the KronEM algorithm, we used the C++ implementation we received from the authors of [10]. We set the algorithm's additional parameters to the values as described in the ReadMe file in their algorithm's distribution.

TABLE II. THE PURITY (ABOVE) AND GED RESULTS (BELOW) FOR THE 2K TEST NETWORKS

Missing Nodes	AA MISC	AA SAMI-A	AA SAMI-N	RCF MISC	RCF SAMI-A	RCF SAMI-N	OASCA	Random
11	0.6610	0.6915	0.6435	0.6740	0.6910	0.6899	0.7077	0.3721
21	0.6246	0.6507	0.6366	0.6363	0.6452	0.6434	0.6525	0.3208
31	0.5914	0.6206	0.6155	0.6079	0.6161	0.6098	0.6214	0.2918
41	0.5703	0.6025	0.5977	0.5838	0.5934	0.5854	0.5944	0.2777
50	0.5472	0.5799	0.5726	0.5647	0.5698	0.5645	0.5727	0.2700
Average	**0.5989**	**0.6291**	**0.6132**	**0.6134**	**0.6231**	**0.6186**	**0.6297**	**0.3065**

Missing Nodes	AA SAMI-A	RCN MISC	RCN SAMI-A	OASCA	KronEM	Random
11	36.55	39.75	37.45	35.35	50.58	70.35
21	71.70	77.00	73.95	73.65	92.66	135.05
31	116.47	123.13	118.18	117.00	138.86	208.90
41	152.02	164.65	159.90	159.05	186.10	277.73
50	190.40	202.38	200.88	199.15	228.64	339.07
Average	**113.43**	**121.38**	**118.07**	**116.84**	**139.37**	**206.22**

VII. RESULTS

A. The Comparison Results

We proceeded to compare the 10 networks of size 2K described above where we randomly removed N missing nodes. In these runs, we used the best weight from the training experiment for each one 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=6 which includes the SAMI-A, SAMI-N and MISC algorithms each with RCN and AA measures. Table II (above) shows the purity results for all of the algorithms (besides KronEM[3]) for each missing nodes option and the overall average result. All of the results are significantly better than the Random algorithm (higher is better). The AA SAMI-A algorithm outperformed all other static algorithms (AA MISC, AA SAMI-N, RCN SAMI-A, etc.). Getting higher results for the OASCA algorithm, with this configuration and without significant difference between AA SAMI-A's performance, confirmed the correctness of the novel measure of relative purity as the basis of the OASCA algorithm. We found that both AA SAMI-A and OASCA results were significantly better than the AA MISC, RCN MISC and AA SAMI-N algorithms (specifically, the ANOVA test of OASCA compared to AA MISC, RCN MISC and AA SAMI-N had a much smaller than 0.05 threshold level with p=2.897E-7, 6.316E-3 and 5.786E-3 respectively). Table II (below) also shows the graph edit distance results for the AA SAMI-A, RCN MISC, RCN SAMI-A, OASCA, KronEM and Random algorithms. Again, all results are significantly better than the Random algorithm (lower is better). We again found that the best results come from using the AA SAMI-A and OASCA algorithms and that these results are also significantly better than the RCN MISC and KronEM algorithms (the ANOVA results for the mean difference of AA SAMI-A compared to RCN MISC and KronEM at the 0.05 significance level being p=2.637E-3 and 1.356E-19, respectively). The RCN MISC results are also significantly better than the KronEM algorithm (ANOVA results for the mean difference being p=9.336E-10).

B. Generalized Configuration

One potential limitation of the algorithms we introduce is the large sizes of their affinity matrices, something that may prevent these algorithms from scaling to networks with larger numbers of nodes. As SAMI-A outperformed SAMI-N, we studied how this algorithm may scale. Furthermore, we wanted to see how well the algorithms do when generalizing from 2K networks (which we used to identify the optimal weights) to larger networks (for which we did not learn the best weights). Our preliminary analysis of the training datasets reduces the attributes affinity matrix M^{Att} from an average of 60%-70% to 7%-15% using a *popularity* threshold=20% and a *noise* threshold=0.15. We run the OASCA algorithm with q=10, considering only the MISC algorithm and the SAMI-A algorithm with several values for the weight parameter w. We use several weight values, as the previously learned weights had been optimized for 2K networks and their values might vary for the larger networks. Specifically, we use the AA MISC and RCN MISC algorithms, 4 weight values for the AA SAMI-A algorithm (w=0.5, 0.6 ,0.7, 0.8) and 4 weight values for the RCN SAMI-A algorithm (w=0.2, 0.3, 0.4, 0.5). We tested the algorithms on 10 instances of 2K, 4K, 8K and 16K networks, with the same 5 missing node values. We repeated the run 5 times over each network.

The results in Table III show that the OASCA algorithm outperforms all other algorithms in all of the tested network sizes. These results were significantly better than the AA SAMI-A algorithm (which gave the second best results) and the original RCN MISC algorithm. Specifically, using the pairwise test at the 0.05 level, we saw that the mean difference of OASCA compared to AA SAMI-A for the 2K, 4K, 8K and 16K networks was p=8.37E-3, 4.90E-6, 3.85E-8 and 5.82E-5 respectively, and compared to RCN MISC was p=3.44E-10, 1.85E-14, 2.13E-12 and 2.27E-9. Getting the best results for the OASCA algorithm shows the effectiveness of the algorithm configuration as well as the correctness of its measure.

TABLE III. THE PURITY RESULTS USING A *popularity* THRESHOLD=20% AND A *nosie* THRESHOLD=0.15

Network Size	AA MISC	AA SAMI-A	RCN MISC	RCN SAMI-A	OASCA	Random
2K	0.5949	0.6184	0.6127	0.6149	0.6237	0.3069
4K	0.5974	0.6223	0.6138	0.6185	0.6313	0.3029
8K	0.5541	0.5759	0.5689	0.5812	0.5920	0.2886
16K	0.5109	0.5404	0.5307	0.5125	0.5475	0.2942

VIII. CONCLUSIONS AND FUTURE WORK

We believe that this paper represents the first work to study 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 how to integrate information about the known nodes to help better solve the missing node problem. Towards this goal, we present two clustering-based algorithms suitable for this problem – SAMI-A and SAMI-N. Both of these algorithms combine the nodes' specific attributes information in two ways. SAMI-A calculates a weighted sum between two *affinity* components, one based on the network graph structure and the other based on specific nodes' attributes. SAMI-N first combines the known nodes' attributes' data into a Social-Attribute Network (SAN) and then uses a weighted sum between different components within the SAN to create the *affinity* measure. We showed that both SAMI-A and SAMI-N

[3]The purity measure is not applicable for the KronEM algorithm as it is not a clustering-based algorithm.

outperform the recent known algorithms, KronEM [10] and MISC [13], which did not use the nodes' specific information.

The second key contribution of this paper is the novel OASCA algorithm, an online algorithm selection for clustering algorithms. We found that even though the best average outcome of AA SAMI−A was significantly higher than all other algorithms, all of the clustering-based algorithms, SAMI−A and SAMI−N, as well as the algorithm on which it was based, MISC, were each best suited for specific problem instances. We also found that parameters within 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 within 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. This allows us to choose, online, the best classifier without running all possibilities in advance, something that was done in previous selection approaches [22], [32], [23] but was not done in this real-world environment.

Our evaluation showed that overall the OASCA algorithm gave the best results. Furthermore, the OASCA algorithm might be suited for any given clustering problem and not only for the missing node identification problem as its input and output do not depend on the problem domain. In the future, we would like to explore additional ways to improve our proposed algorithms' implementation, both SAMI−A and SANI−N, and continue evaluating how these algorithms can be scaled up. We would also like to further explore the potential of the OASCA algorithm with a larger set of algorithms and for large scale networks.

ACKNOWLEDGMENT

This research is based on work supported in part by MAFAT and the Google Interuniversity center for Electronic Markets and Auctions.

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

[15] 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.

[16] ——, "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.

[17] 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.

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

[19] J. Leskovec and C. Faloutsos, "Scalable modeling of real graphs using kronecker multiplication," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 497–504.

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

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

[22] 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.

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

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

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

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

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