2012 IEEE 12th International Conference on Data Mining

Reconstructing Graphs from Neighborhood Data

Dóra Erdös
Boston University
Boston, MA, USA
edori@cs.bu.edu

Rainer Gemulla
Max Planck Institut für Informatik
Saarbrücken, Germany
rgemulla@mpi-inf.mpg.de

Evimaria Terzi
Boston University
Boston, MA, USA
evimaria@cs.bu.edu

***Abstract*—Consider a social network and suppose that we are given the number of common friends between each pair of users. Can we reconstruct the underlying network? Similarly, consider a set of documents and the words that appear in them. If we know the number of common words for every pair of documents, as well as the number of common documents for every pair of words, can we infer which words appear in which documents? In this paper, we develop a general methodology for answering questions like the ones above. We formalize these questions in what we call the RECONSTRUCT problem: Given information about the common neighbors of nodes in a network, our goal is to reconstruct the hidden binary matrix that indicates the presence or absence of relationships between individual nodes. We propose an effective and practical heuristic, which exploits properties of the singular value decomposition of the hidden binary matrix. More specifically, we show that using the available neighborhood information, we can reconstruct the hidden matrix by finding the components of its singular value decomposition and then combining them appropriately. Our extensive experimental study suggests that our methods are able to reconstruct binary matrices of different characteristics with up to 100% accuracy.**

I. INTRODUCTION

The neighbors that are common between a pair of nodes of an undirected graphs carry valuable information about the graph structure. In the context of movie recommendation, for example, we may say that two users are similar if they have watched the same or a largely overlapping set of movies. Likewise, two movies are similar if they have been watched by the same set of users. Such neighborhood information has been exploited successfully in collaborative-filtering algorithms [1], [5], [12], which recommend movies to users based on the number (and ratings) of movies they have in common with other users. As another example, the set of words that are shared between two documents is an indicator of the documents' topical similarity and is exploited by document clustering techniques [1]. Finally, the set of the common friends or common interests between social-network users carry valuable information about the strength and quality of their friendship [9], [13], [21].

Generally, the set of features (e.g., movies, groups, friends, or words) that are shared by two entities (e.g., users or documents) reveal valuable information for data-mining algorithms. Traditionally, this information is extracted directly from the available data, which explicitly states which features are associated with every entity (e.g., movies watched by users, words appearing in a document, friends or interests of a user).

In some cases, however, the original data contains sensitive private information that the dataset owner may not want to share. For example, Netflix may not want to share which customer watched which movie. Similarly Facebook may be unwilling to share the friendship graph or the affiliation graph (i.e., the graph that contains information about the membership of users to groups). In such cases, the data owner can decide to reveal some *aggregate* form of the original data, that preserves enough valuable information for researchers and practitioners to test their data-mining methods, while at the same time hides the characteristics of individual entities.

In this paper, we focus on a particular type of such aggregate information, which we call *neighborhood information*. The neighborhood information of a dataset reveals only the *number* of features shared by every pair of entities (and vice verse), but does not contain information about *which* features are shared. Given such neighborhood information, we try to answer the following question: "To what extent does the revelation of neighborhood information prevent an adversary from reconstructing the original dataset?" For example, in the domain of social networks, the question asks whether or not we can identify the membership of users to groups, given that we we know only the number of common groups between every pair of users and the number of common users for every pair of groups.

We formalize this problem as a bipartite-graph reconstruction problem, which refer to as RECONSTRUCT. Intuitively, RECONSTRUCT assumes a hidden binary dataset that associates entities to features. This dataset can be represented as a bipartite graph, in which an edge between an entity p and a feature q indicates that q is observed in p. The input to the problem is encoded in the following *neighborhood information*: for every pair of entities p and p', we are given the number $\mathbf{L}(e_1, e_2)$ of features shared by the two entities. Similarly, for every pair of features q and q', we are given the number $\mathbf{R}(q, q')$ of entities associated with both features. Given $\mathbf{L}$ and $\mathbf{R}$, our goal is to reconstruct the

1550-4786/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDM.2012.154

hidden bipartite graph.

In this paper, we study different variants of the above RECONSTRUCT problem. Solving the RECONSTRUCT problem is computationally challenging: Our main contribution lies in the design of heuristics that can effectively reconstruct the hidden dataset. The key observation exploited by our heuristics is that we can use the neighborhood information to (approximately) reconstruct parts of the singular value decomposition of the biadjacency matrix of the hidden bipartite graph. We investigate the utility of our methods on a variety of datasets from different application domains. We found that in most cases, the reconstruction error is low; in some cases, our algorithms are able to exactly reconstruct the hidden bipartite graph. We also explore the limits of our approach by hiding some of the entries in matrices $\mathbf{L}$ and $\mathbf{R}$ and provide appropriately modified heuristics that accommodate for such variants. Our experiments indicate that our heuristics can handle missing information to some extent; clearly the reconstruction error increases with the amount of missing neighborhood information.

Roadmap: The rest of the paper is organized as follows: We formally define the RECONSTRUCT problem in Sec. II and give algorithms for solving it in Sec. III. The results of our experimental study are presented in Sec. IV. We discuss related work inand. V and conclude the paper in Sec. VI.

II. PROBLEM DEFINITION

The RECONSTRUCT problem can be expressed both in terms of bipartite graphs as well as binary matrices. Throughout our discussion, we will use both representations interchangeably. We assume that there exists a hidden bipartite graph $G = (P, Q, E)$, where P and Q constitute the sets of nodes in the left and the right partition, respectively. Set $n = |P|$ and $m = |Q$. The edge set $E \subseteq P \times Q$ connects nodes from P with nodes in Q. Every bipartite graph can be represented by its *biadjacency matrix* $\mathbf{M}$. The biadjacency matrix is a binary $n \times m$ matrix with $M(p, q) = 1$ if and only if $(p, q) \in E$. For every node p from P (similarly Q), denote by $N(p)$ the set of neighbors of p in G.

In this paper, we assume that G (and consequently $\mathbf{M}$) is hidden. Our goal is to construct $\mathbf{M}$ from aggregate information. As discussed previously, we focus on the case where the aggregate information consists of the number of common neighbors between all pairs of nodes. Formally, we assume that for each pair $(p, p') \in P \times P$, we are given $\mathbf{L}(p, p') = |N(p) \cap N(p')|$. Similarly, for each pair $(q, q') \in Q \times Q$, we are given $\mathbf{R}(q, q') = |N(q) \cap N(q')|$. We call $\mathbf{L}$ and $\mathbf{R}$ the *neighborhood matrices* of G.

Observe that the main diagonal of the neighborhood matrices contains the degrees of each node. We consider *degree-aware* neighborhood matrices, in which the main diagonals is known, and *degree-oblivious* matrices in which the main diagonal is unknown. We also consider neighborhood matrices in which only parts of the main diagonal are revealed, i.e., only some of the degrees are known. In the course of this paper, we extensively explore the effect of knowledge of node degrees on our estimation problem.

Given $\mathbf{L}$ and $\mathbf{R}$, our goal is to find a binary matrix $\widehat{\mathbf{M}}$ that is as close to $\mathbf{M}$ as possible. Ideally, we aim to minimize the Frobenius norm $F(\widehat{\mathbf{M}}, \mathbf{M}) = \|\widehat{\mathbf{M}} - \mathbf{M}\|_F$, where

$$\|\mathbf{X}\|_F = \sum_{i=1}^{n} \sum_{j=1}^{m} \mathbf{X}(i, j)^2.$$

However, this objective $F(\widehat{\mathbf{M}}, \mathbf{M})$ cannot be computed since $\mathbf{M}$ is unknown (we are given only $\mathbf{L}$ and $\mathbf{R}$). We therefore quantify the quality of $\widehat{\mathbf{M}}$ with respect to $\mathbf{L}$ and $\mathbf{R}$. In more detail, our goal is to minimize the sum of

$$F_{\mathbf{L}}(\widehat{\mathbf{M}}) = \|\widehat{\mathbf{L}} - \mathbf{L}\|_F$$

and

$$F_{\mathbf{R}}(\widehat{\mathbf{M}}) = \|\widehat{\mathbf{R}} - \mathbf{R}\|_F,$$

where $\widehat{\mathbf{R}}$ and $\widehat{\mathbf{L}}$ denote the neighborhood information induced by $\widehat{\mathbf{M}}$ (see Sec. III for details).

*Problem 1 (*RECONSTRUCT*):* Given neighborhood matrices $\mathbf{L}$ and $\mathbf{R}$, find a binary matrix $\widehat{\mathbf{M}}$ that minimizes the sum $F_{\mathbf{L}}(\widehat{\mathbf{M}}) + F_{\mathbf{R}}(\widehat{\mathbf{M}})$.

In case matrices $\mathbf{L}$ and $\mathbf{R}$ are degree-oblivious, we modify the above definitions such that the main diagonals are not taken into account. Since the type of the neighborhood matrices will always be clear from the context, we abuse notation and write $F_{\mathbf{L}}(\widehat{\mathbf{M}})$ and $F_{\mathbf{R}}(\widehat{\mathbf{M}})$ for both degree-aware and degree-oblivious matrices. In what follows, we write RECONSTRUCT-DA to refer to the degree-aware case, and RECONSTRUCT-DO to refer to the degree-oblivious variant.

Discussion: The above problem definitions as well as the algorithms presented in the next section also apply to general (non-bipartite) graphs: We simply use the graph's adjacency matrix instead of its biadjacency matrix (and redefine $\mathbf{L}$ and $\mathbf{R}$ appropriately). In fact, our algorithms are oblivious to the fact that the hidden matrix constitutes a biadjacency matrix of some graph, i.e., they apply to any binary matrix. In what follows, we focus on bipartite graphs for clarity of exposition.

III. ALGORITHMS

In this section, we present algorithms for solving the RECONSTRUCT problem. The high-level idea of our algorithms is to compute $\widehat{\mathbf{M}}$ by reconstructing the components of the *singular value decomposition* (SVD) of $\mathbf{M}$. Before describing our algorithms in detail, we provide some background on the *eigendecomposition* and the *singular value decomposition*. See [8] for an in-depth treatment of these decompositions.

A. Matrix Decompositions

The *eigendecomposition* of an arbitrary *symmetric* matrix $\mathbf{X}$ is a decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where $\mathbf{U}$ is a unitary matrix having as columns the normalized eigenvectors of $\mathbf{X}$, and $\mathbf{\Lambda}$ is a diagonal matrix containing the corresponding eigenvalues of $\mathbf{X}$.

The *singular value decomposition (SVD)* of an arbitrary matrix $\mathbf{X}$ is a decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U}$ (resp. $\mathbf{V}$) is an orthonormal matrix with the left (resp. right) singular vectors of $\mathbf{X}$ as its columns, and $\mathbf{\Sigma}$ is a diagonal matrix that contains the singular values of $\mathbf{X}$ in its main diagonal.

When matrix $\mathbf{X}$ is symmetric and positive semi-definite, then the left and the right singular vectors are equal. In this case, the eigendecomposition and the SVD decomposition coincide. The following fact is known about the SVD decompositions of a matrix $\mathbf{X}$ as well as matrices $\mathbf{X}\mathbf{X}^T$ and $\mathbf{X}^T\mathbf{X}$.

Proposition 1: If matrix $\mathbf{X}$ is an $n \times m$ matrix and its SVD is $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, then the SVD decomposition of $\mathbf{X}\mathbf{X}^T$ is

$$\mathbf{X}\mathbf{X}^T = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^T \tag{1}$$

and the SVD decomposition of $\mathbf{X}^T\mathbf{X}$ is

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T. \tag{2}$$

To see this, substitute $\mathbf{X}$ with $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ in the left-hand-side of Eq. (1) to obtain

$$\begin{aligned}\mathbf{X}\mathbf{X}^T &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right)^T \\ &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^T.\end{aligned}$$

The proof of Eq. (2) is similar and omitted.

Moreover, it is known that a truncated SVD can give the best low-rank approximation of $\mathbf{X}$.

Proposition 2 (Eckart and Young [7]): If $\mathbf{U}_k$ (resp. $\mathbf{V}_k$) represents the left (resp. right) singular vectors that correspond to the k singular values $\mathbf{\Sigma}_k$ of the largest magnitude, then matrix $\mathbf{X}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$ is the best rank-k approximation of $\mathbf{X}$ in terms of the Frobenius norm. That is, $\mathbf{X}_k$ minimizes

$$\|\mathbf{X} - \mathbf{X}_k\|_F = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(\mathbf{X}(i,j) - \mathbf{X}_k(i,j)\right)^2.$$

Observe, that we can write $\mathbf{X}_k$ as the sum of k rank-1 matrices:

$$\mathbf{X}_k = \sum_{i=1}^{k}\mathbf{U}(:,i)\mathbf{\Sigma}(i,i)\mathbf{V}(:,i)^T. \tag{3}$$

Here $\mathbf{X}(:,i)$ denotes the i-th column of $\mathbf{X}$. The decomposition of $\mathbf{X}_k$ into a sum of rank-1 matrices turns out to be useful for our estimation algorithms since it allow us to determine the elements of the singular value decomposition one component at a time.

B. Solving the RECONSTRUCT-DA Problem

Recall that in the RECONSTRUCT-DA problem, we assume that the diagonal elements of the neighborhood matrices are equal to the degree of the nodes in the hidden bipartite graph. We refer to such neighborhood matrices by $\mathbf{L}_d$ and $\mathbf{R}_d$. The key to our approach for RECONSTRUCT-DA is the following observation.

Observation 1: The matrices $\mathbf{L}_d$ and $\mathbf{R}_d$ are given by $\mathbf{L}_d = \mathbf{M}\mathbf{M}^T$ and $\mathbf{R}_d = \mathbf{M}^T\mathbf{M}$.

This observation connects the hidden data matrix with the observed neighborhood matrices. It allows us to use the singular value decomposition to devise an efficient heuristic algorithm.

Denote by $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ the the SVD of the *unknown* matrix $\mathbf{M}$. Combining Observation 1 with Proposition 1, we obtain $\mathbf{L}_d = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ and $\mathbf{R}_d = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$. This means that the eigendecompositions of $\mathbf{L}_d$ and $\mathbf{R}_d$ provide the left and the right singular vectors of $\mathbf{M}$. Additionally, we obtain $|\mathbf{\Sigma}| = \sqrt{\mathbf{\Lambda}}$, where both the absolute value and square root is taken element-wise. Intuitively, our goal is to exploit this knowledge to reconstruct the unknown matrix $\mathbf{M}$. However, it is not immediately clear how to proceed since we know only the absolute value but not the sign of the diagonal elements in $\mathbf{\Sigma}$.

More formally, set $\sigma_i = \mathbf{\Sigma}(i,i)$ and $\lambda_i = \mathbf{\Lambda}(i,i)$. We assume without loss of generality that the singular values are reported in decreasing order of magnitude, i.e., $|\sigma_1| \geq |\sigma_2| \geq \ldots \geq |\sigma_n|$. By Proposition 1, we know that $\lambda_i = \sigma_i^2$ and thus $\sqrt{\lambda_i} = |\sigma_i|$. This means that σ_i can take two possible values: $-\sqrt{\lambda_i}$ and $\sqrt{\lambda_i}$. Let $\widehat{\mathbf{\Sigma}}$ be a diagonal matrix with values $\widehat{\sigma_1}, \ldots, \widehat{\sigma_n}$ in its main diagonal, such that $|\widehat{\sigma_i}| = |\sigma_i|$. Here each $\widehat{\sigma_i}$ is signed, that is, it is either positive or negative. We refer to $\widehat{\mathbf{\Sigma}}$ as a *sign assignment* of $\mathbf{\Sigma}$. Given a sign assignment $\widehat{\mathbf{\Sigma}}$, matrix $\widehat{\mathbf{M}} = \mathbf{U}\widehat{\mathbf{\Sigma}}\mathbf{V}^T$ constitutes an *estimate* of $\mathbf{M}$. Note that $\widehat{\mathbf{M}}$ may not be a binary matrix; we return to this issue below.

Our algorithms aim to find the "best" sign assignment, i.e., the one that produces the best estimate of $\mathbf{M}$. In order to do this, we need to address the following two questions: (a) How do we evaluate a given sign assignment and (b) how can we find good sign assignments?

Evaluating a sign assignment. Given $\mathbf{U}$ and $\mathbf{V}$ together with a sign assignment $\widehat{\mathbf{\Sigma}}$, we want to decide how well we can reconstruct $\mathbf{M}$. Ideally, we would like to evaluate $\widehat{\mathbf{\Sigma}}$ by computing $F(\widehat{\mathbf{M}}, \mathbf{M})$. Unfortunately, this approach is infeasible because $\mathbf{M}$ is unknown. Alternatively, we could try to set $\widehat{\mathbf{M}} = \mathbf{U}\widehat{\mathbf{\Sigma}}\mathbf{V}^T$ and compute $F_{\mathbf{L}} + F_{\mathbf{R}}$. However, this approach is also not helpful because the quantities $F_{\mathbf{L}}$ and $F_{\mathbf{R}}$ do not depend on the sign assignment. To see this, observe that the elements of $\mathbf{\Sigma}$ get squared when we compute $\widehat{\mathbf{L}} = \widehat{\mathbf{M}}\widehat{\mathbf{M}}^T$ and $\widehat{\mathbf{R}} = \widehat{\mathbf{M}}^T\widehat{\mathbf{M}}$. We propose a way to evaluate the sign assignment $\widehat{\mathbf{\Sigma}}$, which utilizes the fact that $\mathbf{M}$ is a *binary* matrix: If $\widehat{\mathbf{M}}$ is a good estimate of $\mathbf{M}$,

then it is close to $\mathbf{M}$ and—as a result—close to a *binary* matrix. Let $\mathbf{X}$ be an arbitrary matrix and define its *binary counterpart* $\mathbf{BX}$ to be the binary matrix closest to $\mathbf{X}$ in terms of the Frobenius norm:

$$\mathbf{BX}(i,j) = \begin{cases} 1 & \text{if } \mathbf{X}(i,j) > 0.5 \\ 0 & \text{else} \end{cases}$$

Clearly good estimates of $\widehat{\mathbf{M}}$ must be close to their binary counterparts. The opposite may or may not hold, i.e., an estimate $\widehat{\mathbf{M}}$ that is close to its binary counterpart may or may not be close to $\mathbf{M}$. Nevertheless, our experiments give strong evidence that minimizing $F(\widehat{\mathbf{X}}, \mathbf{B}\widehat{\mathbf{X}})$ leads to a low reconstruction error. Although we do not make any formal claims, our hope is that there are few (or ideally just one) sign assignments that produce a binary matrix $\widehat{\mathbf{M}}$.

The `Greedy` **sign assignment algorithm.** The `Greedy` algorithm takes as input matrices $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}$ and outputs an estimate $\widehat{\mathbf{M}}$. The algorithm constructs $\widehat{\mathbf{M}}$ in k iterations, where k is a parameter that influences accuracy. In iteration i, `Greedy` determines the sign of σ_i. In more detail, we compute matrices $\mathbf{M}_i^+$ and $\mathbf{M}_i^-$ by considering the positive and negative sign for σ_i, respectively. The signs of of $\sigma_1, \ldots, \sigma_{i-1}$ are taken from previous iterations, and $\sigma_{i+1}, \ldots, \sigma_n$ are taken to be zero. We then set the sign of the i-th singular value based on whether $\mathbf{M}_i^+$ or $\mathbf{M}_i^-$ is closer to its binary counterpart. The intuition behind this approach is that large singular values, which are processed first, have significant impact on the estimate $\widehat{\mathbf{M}}$ so that we expect `Greedy` to make correct decisions. After k iterations have been completed, our algorithm produces a sign assignment $\widehat{\mathbf{\Sigma}_k}$ for the k singular values of the largest magnitude. Algorithm 1 gives pseudo code for `Greedy`. Here we compute $\mathbf{M}_i^+$ and $\mathbf{M}_i^-$ from $\mathbf{M}_{i-1}^+$ and $\mathbf{M}_{i-1}^-$, respectively, for improved efficiency.

Algorithm 1 `Greedy` algorithm to compute an optimal sign assignment $\widehat{\mathbf{\Sigma}}$ and construct $\widehat{\mathbf{M}}$.

Input: $\mathbf{U}_k$, $\mathbf{\Sigma}_k$, $\mathbf{V}_k$ and integer k
Output: $\widehat{\mathbf{M}}_k$
$\widehat{\mathbf{M}}_0 \leftarrow 0_{n \times m}$
for $i = 1 \ldots k$ **do**
  $\mathbf{M}_i^+ = \widehat{\mathbf{M}}_{i-1} + \mathbf{U}_k(:,i)\sigma_i\mathbf{V}_k(:,i)^T$
  $\mathbf{M}_i^- = \widehat{\mathbf{M}}_{i-1} - \mathbf{U}_k(:,i)\sigma_i\mathbf{V}_k(:,i)^T$
  if $F(\mathbf{M}_i^+, \mathbf{BM}_i^+) < F(\mathbf{M}_i^-, \mathbf{BM}_i^-)$ **then**
    $\widehat{\mathbf{M}}_i = \mathbf{M}_i^+$
  else
    $\widehat{\mathbf{M}}_i = \mathbf{M}_i^-$
  end if
end for
return $\widehat{\mathbf{M}}_k$

The `RecSVD` **algorithm.** We can use the `Greedy` algorithm to solve the RECONSTRUCT-DA problem as follows: We first compute the eigenvectors $\mathbf{U}$ and $\mathbf{V}$ as well as the eigenvalues $\mathbf{\Lambda}$ of both $\mathbf{L}$ and $\mathbf{R}$. Note that $\mathbf{\Lambda}$ is the same in both eigendecompositions. We then input matrices $\mathbf{U}$, $\sqrt{\mathbf{\Lambda}}$, and $\mathbf{V}$ to the `Greedy` algorithm to construct $\widehat{\mathbf{M}}$, which is subsequently rounded to a binary matrix. We refer to this complete algorithm as `RecSVD`. Given a value of k, `RecSVD` proceeds as follows:

- Compute the truncated SVD of $\mathbf{L}_d$ and $\mathbf{R}_d$
  1: $\mathbf{L}_d \approx \mathbf{U}_k\mathbf{\Lambda}_k\mathbf{U}_k^T$
  2: $\mathbf{R}_d \approx \mathbf{V}_k\mathbf{\Lambda}_k\mathbf{V}_k^T$
- Run `Greedy`$(\mathbf{U}_k, \sqrt{\mathbf{\Lambda}_k}, \mathbf{V}_k, k)$ to obtain $\widehat{\mathbf{M}}_k$
- Output $\mathbf{B}\widehat{\mathbf{M}}_k$

Observe that `RecSVD` does not use all the of left and the right singular vectors obtained by the eigendecompositions of $\mathbf{L}_d$ and $\mathbf{R}_d$ but only the ones with the k eigenvalues of largest magnitude. For this reason, we compute only the truncated SVD of $\mathbf{L}_d$ and $\mathbf{R}_d$ (see also Proposition 2), which significantly reduces computational costs.

Discussion. The impact of the number k of singular values on the reconstruction problem is explored extensively in the experimental section of this paper. To give a preview, we found that `RecSVD` performs extremely well in practice: It can reconstruct binary matrices with thousands of rows and columns almost perfectly with as few as 300 singular values. To provide some insight into the performance of `RecSVD`, consider the case in which the hidden data matrix $\mathbf{M}$ is block-diagonal with at most k blocks, and that each block consists of only 1s. Then the rows and columns of $\mathbf{M}$ can be grouped into linearly independent groups, each described by one pair of singular vectors and a singular value. In such matrices, the `RecSVD` algorithm will optimally reconstruct $\mathbf{M}$. In general, we expect the estimate $\widehat{\mathbf{M}}$ produced by `RecSVD` to be closer or further away from the true matrix $\mathbf{M}$ depending on how "clear" the block structure in $\mathbf{M}$ is (after appropriate permutation of rows and columns).

Running time of `RecSVD`**.** Assume w.l.o.g. that $n \geq m$. Since we we only compute the k largest-magnitude eigenvalues (and their corresponding eigenvectors) of $\mathbf{L}_d$ and $\mathbf{R}_d$, the running time of the eigendecomposition is $O(n^2k)$. The running time of the `Greedy` routine is $O(n^2k)$ as well so that the total time complexity of `RecSVD` is $O(n^2k)$.

Speeding up `RecSVD`**.** Since eigendecompositions are useful for many problems, there exists a vast majority of work devoted to speed up these computations (see, for example, the sampling-based approach in [6]). Such methods can be utilized—whenever needed—to speed up the first step of the `RecSVD` algorithm.

In what follows, we describe a technique to speed up the second step of `RecSVD`, i.e., the `Greedy` algorithm given in

Algorithm 1. Observe that the computations done in lines 3, 4, and 5 of Algorithm 1 require $O(n^2k)$ time. We can speed up this computation significantly by sampling the rows of $\mathbf{U}_k$ and $\mathbf{V}_k$. If we use a sample of c rows for some constant c, the running time is reduced to $O(c^2k)$. In order to choose the rows that affect the entries in $\widehat{\mathbf{M}}$ the most, we sample row r with probability proportional to $\sum_{i=1}^{k} |\mathbf{U}_k(r,i) \cdot \mathbf{V}_k(r,i)|$. We refer to this sampling algorithm as `Greedy_S`. In our experiments, we found that samples of size as small as $c = 300$ provide almost identical results to `Greedy`. Note that we use sampling in `Greedy_S` only to determine the sign assignment $\widehat{\mathbf{\Sigma}_k}$. Once $\widehat{\mathbf{\Sigma}_k}$ is determined, we compute $\widehat{\mathbf{M}}_k$ using the entire matrices $\mathbf{U}_k$ and $\mathbf{V}_k$.

C. Solving the RECONSTRUCT-DO Problem

In this section, we turn our attention to the RECONSTRUCT-DO problem. Recall that in the RECONSTRUCT-DO problem, we want to estimate $\widehat{\mathbf{M}}$ from degree-oblivious matrices $\mathbf{L}$ and $\mathbf{R}$. In this case, the degrees of nodes in P and Q are unknown, i.e, the main diagonal elements of $\mathbf{L}$ and $\mathbf{R}$ are all zero.

We solve the RECONSTRUCT-DO problem with an iterative version of `RecSVD`. We call this new routine `RecSVD-iter`. The `RecSVD-iter` algorithm starts with some initialization of the diagonal matrices $\widehat{\mathbf{D}}_P$ and $\widehat{\mathbf{D}}_Q$ corresponding to the estimated node degrees in P and Q, respectively; we discuss various choices of initial values below. In every iteration, we run `RecSVD` using input matrices $\widehat{\mathbf{L}}_d = \mathbf{L} + \widehat{\mathbf{D}}_P$ and $\widehat{\mathbf{R}}_d = \mathbf{R} + \widehat{\mathbf{D}}_Q$. Afterwards, we revise the node degrees by performing an educated guess of new values for $\widehat{\mathbf{D}}_P$ and $\widehat{\mathbf{D}}_Q$. The new values are based on the output matrix $\mathbf{B}\widehat{\mathbf{M}}$ of `RecSVD`. In more detail, we first compute the updated versions of $\widehat{\mathbf{L}}_d = (\mathbf{B}\widehat{\mathbf{M}})(\mathbf{B}\widehat{\mathbf{M}})^T$ and $\widehat{\mathbf{R}}_d = (\mathbf{B}\widehat{\mathbf{M}})^T(\mathbf{B}\widehat{\mathbf{M}})$. Then new estimates of $\widehat{\mathbf{D}}_P$ and $\widehat{\mathbf{D}}_Q$ are obtained by the diagonal entries of $\widehat{\mathbf{L}}_d$ and $\widehat{\mathbf{R}}_d$, respectively.

Observe that in `RecSVD-iter`, we use only the main diagonals of $\widehat{\mathbf{L}}_d$ and $\widehat{\mathbf{R}}_d$ in order to update $\widehat{\mathbf{D}}_P$ and $\widehat{\mathbf{D}}_Q$. Hence we can speed up computation by computing only the diagonals of $\widehat{\mathbf{L}}_d$ and $\widehat{\mathbf{R}}_d$.

Scheinerman and Tucker [18] use a similar iterative approach to compute the eigenvalue decomposition of symmetric matrices with missing entries. They also show that convergence of the diagonals $\widehat{\mathbf{D}}_P$ and $\widehat{\mathbf{D}}_Q$ is not guaranteed. In practice, however, convergence is fast in most cases. In our experience, we found that 100 iterations sufficed to achieve convergence on all our datasets.

Running time of `RecSVD-iter`. Every iteration of `RecSVD-iter` performs a call to `RecSVD`. Under the assumption that $n > m$, we obtain $O(n^2k)$ time per iteration. Using the optimization mentioned above, the computation of $\widehat{\mathbf{D}}_P$ and $\widehat{\mathbf{D}}_Q$ takes at most $O(n^2)$ time. When we run `RecSVD-iter` for t iterations, we obtain a total time complexity of $O(n^2kt)$.

Partial knowledge of the degrees. The `RecSVD-iter` algorithm is an iterative algorithm that tries to guess the degrees of the nodes in P and Q before reconstructing the unknown bipartite graph. One of the main characteristics of `RecSVD-iter` is that it can also work when only some of the entries of the degrees of the nodes in P or Q are known. In these cases, the algorithm will simply iterate in order to obtain estimates of the unknown degrees only. The ability of the algorithm to accept some of these entries as input and to guess the rest allows us to explore the extent to which knowledge of the nodes' degrees improves our ability to reconstruct the hidden matrix. Our experimental results demonstrate that knowledge of even a small fraction of the high degree nodes can considerably affect reconstruction quality.

Initializing $\widehat{\mathbf{D}}_P$ and $\widehat{\mathbf{D}}_Q$. The perhaps simplest way to initialize $\widehat{\mathbf{D}}_P$ and $\widehat{\mathbf{D}}_Q$ is to set the main diagonal to zero; we refer to this strategy as `zero`. We can, however, start with a better initial guess: Observe that a lower bound of the values in the main diagonal is given by the maximum values of every row of $\mathbf{L}$ and $\mathbf{R}$. An upper bound is obtained similarly. We refer to the heuristics that use the lower and upper bounds as initial guesses as `lower` and `upper` respectively.

IV. EXPERIMENTS

In this section, we describe the results of an extensive experimental study on both synthetic and real-world datasets.

A. Experimental Setup

We used a 12-core machine with Intel X5650 2.67GHz CPUs and 2GB of memory per core. All algorithms were implemented in Matlab.

Evaluation metrics. We report results with regard to two different error metrics. The first metric one is the *relative Frobenius error* (RFE) given by

$$\text{RFE} = \frac{F_{\mathbf{L}}(\mathbf{B}\widehat{\mathbf{M}}) + F_{\mathbf{R}}(\mathbf{B}\widehat{\mathbf{M}})}{\|\mathbf{L}_d\|_F + \|\mathbf{R}_d\|_F}.$$

Observe that $F_{\mathbf{L}}(\mathbf{B}\widehat{\mathbf{M}}) + F_{\mathbf{R}}(\mathbf{B}\widehat{\mathbf{M}}) = 1$ when $\widehat{\mathbf{M}}$ has all its entries equal to zero. Thus RFE expresses the improvement over the all-zero solution to RECONSTRUCT with regard to the neighborhood information. Note that the sign assignment $\widehat{\Sigma}$ does affect the RFE since we use the binary counterpart $\mathbf{B}\widehat{\mathbf{M}}$ instead of using the real matrix $\widehat{\mathbf{M}}$ (cf. Sec. III).

The second metric is the *relative absolute error* (RAE) measured with respect to the ground truth $\mathbf{M}$. It is given by

$$\text{RAE} = \frac{\|\mathbf{B}\widehat{\mathbf{M}} - \mathbf{M}\|_1}{\|\mathbf{M}\|_1},$$

where $||\mathbf{X}||_1 = \sum_{ij} |\mathbf{X}(i,j)|$. The RAE measures the number of incorrect entries in estimate $\mathbf{B}\widehat{\mathbf{M}}$ relative to the number of non-zeros in $\mathbf{M}$. Thus the all-zero solution obtains an RAE of 1.

Synthetic datasets. As mentioned in Section III-B, our algorithms explore the underlying block-structure in the data matrix, if existent. For this reason, we experimented with synthetic block-diagonal binary matrices. We first created a set of rectangular blocks of all 1s with sizes chosen uniformly and at random in between 1 and 100. The blocks are then arranged to form a block-diagonal matrix. This process generates a 0/1 matrix with only 1s appearing in each block. In order to evaluate the ability of our algorithm to handle noise, we also randomly flip 10% of the zeros to ones, and 10% of the ones to zeros. We refer to the resulting datasets as BLOCK; the particular instance used in our experiments contains 45 blocks and has size 2400×1000.

Real-life datasets. We perform experiments on two real-life datasets, which are examples of the applications mentioned in Section I. Even though these datasets do not have a distinct block-diagonal structure, we found that our algorithms produce low-error reconstructions.

The FLICKR dataset contains information from the photo sharing site `www.flickr.com`; it was provided by Zheleva *et al.* [20]. Users of Flickr can form groups based on common interests. A user u is connected by an edge to group g if u is a member in g. We randomly sampled 2000 users and 1989 groups from the original data. The resulting bipartite graph contains about 2×10^6 edges.

The CORA dataset consists of 2708 scientific publications and a dictionary of 1433 unique words. The dataset contains occurrences of words in the documents: We set $\mathbf{M}(d, w) = 1$ in the biadjacency matrix if document d contains word w. The dataset only contains words which have document frequency at least 10. The resulting bipartite graph contains about 50K edges.[1]

Methodology. For all of our datasets, we use the respective biadjacency matrices to extract the neighborhood matrices $\mathbf{L}_d$ and $\mathbf{R}_d$ (resp. $\mathbf{L}$ and $\mathbf{R}$). The matrices are used as input to our algorithms for RECONSTRUCT-DA (resp. RECONSTRUCT-DO).

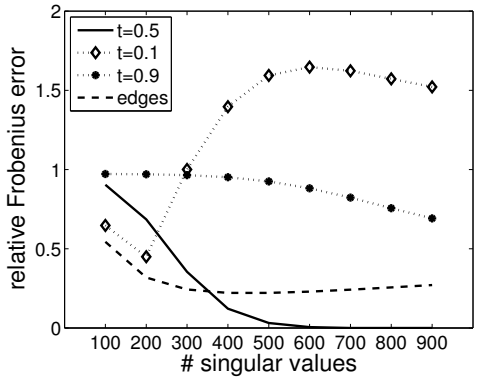
B. Results for RECONSTRUCT-DA

In Figure 1, we report the performance of `RecSVD` on the BLOCK dataset. Figure 1(a) shows the RFE and Figure 1(b) the RAE as a function of the number k of singular values. The curves for $t = 0.5$, $t = 0.1$ and $t = 0.9$ correspond to different decision thresholds applied when computing the binary counterpart of $\widehat{\mathbf{M}}$. Given a decision threshold t, we define the binary counterpart $\mathbf{B}_t\mathbf{X}$ as

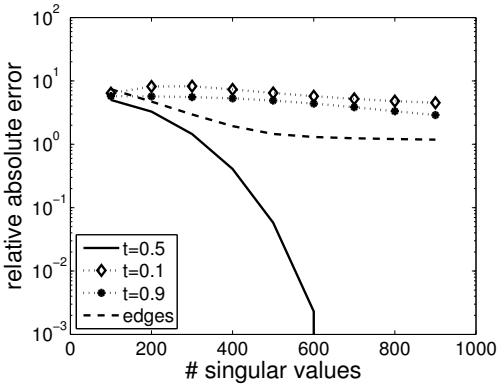
$$\mathbf{B}_t\mathbf{X}(i,j) = \begin{cases} 1 & \text{if } \mathbf{X}(i,j) > t \\ 0 & \text{else} \end{cases}$$

In contrast, method `#edges` does not use a prespecified threshold but rather sets to 1 the top-e entries of $\widehat{\mathbf{M}}$, where

[1]The CORA dataset is available at: http://www.cs.umd.edu/~sen/lbc-proj/LBC.html.



(a) Relative Frobenius error



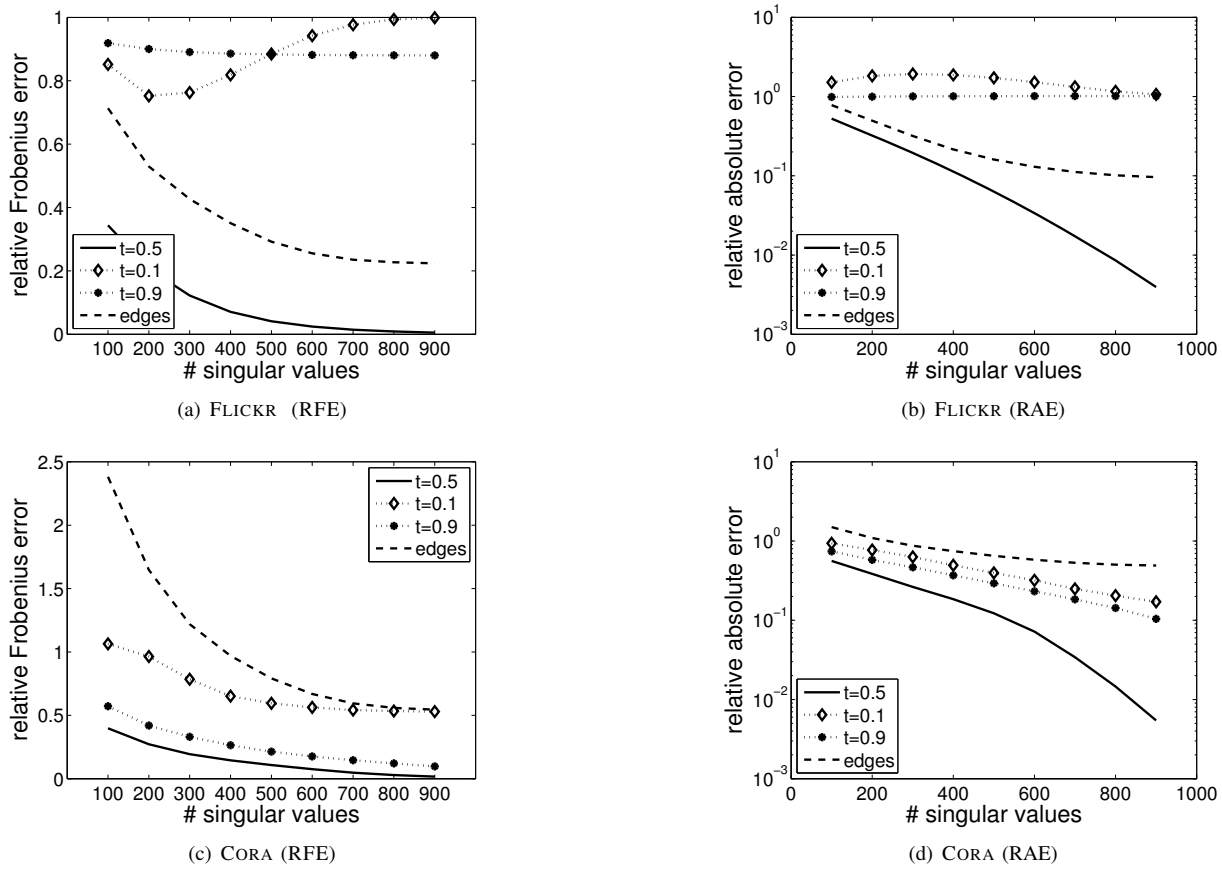
(b) Relative absolute error

Figure 1: Performance of the `RecSVD` algorithm on the BLOCK dataset.

e is the number of edges in the hidden binary matrix. Note that e is available for RECONSTRUCT-DA: Its value can be computed by the trace of either $\mathbf{L}_d$ or $\mathbf{R}_d$. We modify the `RecSVD` algorithm such that it uses the binary counterparts as defined above.

From Figures 1(a) and 1(b), we can see that 600 singular values are sufficient to perfectly reconstruct the BLOCK dataset with $t = 0.5$. This shows that the binary rounding heuristic applied in `RecSVD` can reconstruct a full-rank matrix from a truncated SVD decomposition. This result is possible because the *effective* rank of BLOCK is low (about 45). Note that the performance of our algorithms degrades if we use decision thresholds different from $t = 0.5$ or the `#edges` heuristic. Thus our choice $\mathbf{B}\widehat{\mathbf{M}}$ given in Sec. III appears to work best in practice.

Figure 2 shows our results for the FLICKR and CORA datasets using `RecSVD`. Figures 2(a) and (c) report the RFE of the reconstruction, whereas Figures 2(b) and (d) report the corresponding RAE. As before, the `RecSVD` algorithm with threshold $t = 0.5$ performs the best across the board, achieving close to zero error in terms of both RFE and RAE. We want to draw attention to the RAE results on



(a) FLICKR (RFE) (b) FLICKR (RAE)

(c) CORA (RFE) (d) CORA (RAE)

Figure 2: Performance of the `RecSVD` algorithm for RECONSTRUCT-DA on the FLICKR and CORA datasets.

the FLICKR data in Figure 2(b). The results exhibit a very clean separation between the performance of `RecSVD` for $t = 0.5$, `#edges`, and `RecSVD` for $t = 0.1$ and $t = 0.9$. This is natural in the sense that $t = 0.1$ and 0.9 provide an estimate matrix of (almost) all 1s (for $t = 0.1$) or all 0s (for $t = 0.9$). The reason that this difference is so pronounced on the FLICKR dataset is due to the fact that the degree distribution of the nodes in this graph fit very well to a power law distribution. Thus, predicting all zeros or all ones are not good guesses for the underlying hidden matrix.

In Figure 3, we report the performance of `RecSVD` on the CORA dataset, where we used the sampling-based greedy approach `Greedy_S`. In the figure, we give the RFE achieved by `Greedy_S` as a function of the number of singular values. The different curves in the figure correspond to the different sample sizes. We observe that even a sample of $c = 100$ rows of $\mathbf{U}$ and $\mathbf{V}$ gave reasonable results. When we used a sample size of $c = 300$, we obtained identical results to `RecSVD` run without sampling. To put this into perspective, recall that the size of the CORA dataset is 2708×1433. The `RecSVD` algorithm with the `Greedy_S` heuristic performed similarly well on the other two datasets (not shown here). Besides the good performance with respect to RFE, the sampling approach achieves a more than 15-times speedup per iteration w.r.t. the `Greedy` algorithm for $c = 300$ samples.

C. *Results for* RECONSTRUCT-DO

We proceed to report our results for RECONSTRUCT-DO. In all experiments with the `#edges` heuristic, we provided the number e of edges as additional input.

Although `RecSVD` is very effective on all our datasets for RECONSTRUCT-DA, the performance of the `RecSVD-iter` for RECONSTRUCT-DO problem is less so. Figure 4 shows the performance of `RecSVD-iter` with respect to RAE on the BLOCK dataset. The RAE achieved on the RECONSTRUCT-DO problem for 900 singular values is given by ≈ 0.11, which is reasonably low. A deeper analysis of this result showed that the error is roughly equally distributed in terms of misclassified 0s and 1s: 132112 out of the 2033248 ones in the dataset are missed, whereas 97284 zero entries are incorrectly predicted as 1s. This is in contrast to the RECONSTRUCT-DA results of Figure 1(b), where we achieved an RAE of 0. This difference demonstrates that the

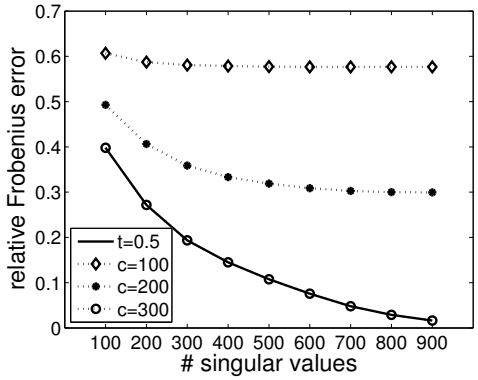


Figure 3: Relative Frobenius error of the `RecSVD` algorithm with the `Greedy_S` sampling heuristic on the CORA dataset. The curves for `c=300` and `t=0.5` coincide.

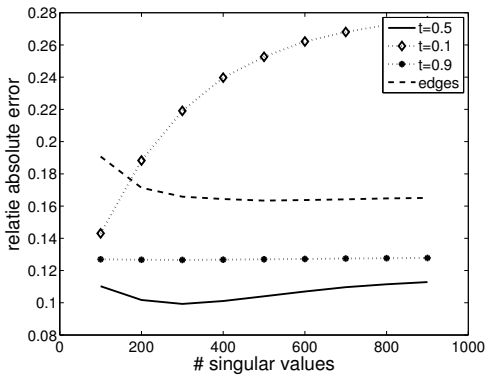


Figure 4: Relative absolute error of `RecSVD-iter` on BLOCK data (`zero` initialization).

node degrees carry valuable information; a complete lack of this knowledge considerably degrades reconstruction quality.

We also experimented with the `lower` and `upper` heuristics for initializing the degrees. Additionally, we tried the following two initializations:

`top`: In this heuristic we run `RecSVD-iter` using as input the degrees of the highest 0%, 25%, 50% , 75% or 100% degree nodes (all other degrees are initialized to 0). During the iterative step of the `RecSVD-iter` algorithm, we fix these values and only update the remaining part of the diagonals.

`rand`: We also evaluated the performance of `RecSVD-iter` when the degrees of arbitrary nodes are known (as opposed to the highest-degree ones). We fixed a randomly-chosen set of 0%, 25%, 50% , 75%, or 100% of the degrees in the main diagonal (again, the remaining degrees are initialized to 0).

The performance of these initialization methods for the CORA and the FLICKR datasets is shown in Figures 5(a) and 5(b) respectively. Here we focus on the RAE only, set $t = 0.5$ and used $k = 900$ singular values. In each plot, the x-axis corresponds to the percentage of the known degrees in the main diagonals.

First, note that `lower` and `upper` do not depend on any known values in the main diagonal, thus the results are the same for all percentages; we only report them to make visual comparison easier. Second, note that 0% corresponds to the completely hidden diagonal while 100% implies that the diagonal is fully known so that results are identical to the performance of `RecSVD` on RECONSTRUCT-DA. Moreover, `top` and `rand` perform identically for 0% and for 100% since either no or all nodes are selected.

As we can see, neither `rand`, nor `lower`, nor `upper` perform well: their RAE is consistently above 0.5 and in some cases even worse the baseline of the all-zero solution. Note that `zero` initialization performed similarly: We achieved an RAE of around 0.6 and 0.9 for the FLICKR and the CORA datasets, respectively. In contrast, `top` works much better. Interestingly, knowing the highest 25% of degrees performs as well as knowing the highest 75%. This is due to the fact that larger degree nodes imply a denser row in the underlying $\mathbf{M}$, which in turn corresponds to the largest singular values of $\mathbf{M}$. The smaller degrees only affect the smaller magnitude singular values in the SVD so that the error introduced by not knowing them is absorbed by the binary rounding in the `Greedy` algorithm.

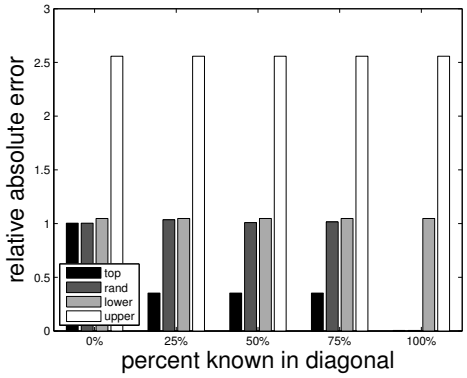
D. Computational Cost

Recall that we need to (1) compute the truncated singular value decomposition of $\widehat{\mathbf{M}}$ and (2) compute the sign assignment $\widehat{\mathbf{\Sigma}}$. The cost of computing the truncated SVD depends on the size of the dataset. It took 0.056 secs per singular value for CORA and 0.092 secs per singular value for FLICKR. Running `Greedy` in the `RecSVD` algorithm took 0.18 secs per singular value for CORA and 0.21 secs for FLICKR. The `Greedy_S` algorithm in the speedup version of our algorithm achieved a 15-fold speedup over `Greedy` for CORA with a sample of size $c = 300$.

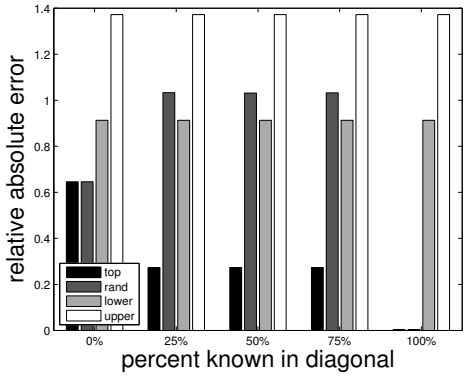
V. RELATED WORK

To the best of our knowledge, the problem of reconstructing binary matrices from neighborhood information has not been addressed before. However, our work is related to previous work on the analysis of real and binary matrices, as summarized below.

Matrix reconstruction. Existing work on matrix reconstruction focuses on the reconstruction of real-valued matrices from a few or noisy entries [3], [10]. Such reconstruction problems, also known as *matrix-completion* problems, have received a lot of attention over the last years and existing studies have led to interesting algorithmic results. Although the goal of these methods are similar in spirit to our work



(a) CORA



(b) FLICKR

Figure 5: Performance of `RecSVD-iter` combined with `rand`, `top`, `lower` and `upper` and a varying faction of revealed entries on the main diagonal.

in that they aim to reconstruct a hidden data matrix, they are tailored to real-valued matrices. In contrast, our work focuses on the reconstruction of binary matrices. More importantly, matrix-completion methods use information about the values of a subset of the hidden matrix in order to reconstruct it. Such information is not available in our problem, in which only the per-node neighborhood information but no information about individual elements is provided.

Binary reconstruction. The problem of reconstructing binary matrices also arises in computer vision, for example. Here the goal is to reconstruct a noise-free image from a noisy version of a black and white image. Existing methods for these problems either use combinatorial [11] or statistical inference techniques [2]. These methods rely on the fact that a noisy version of the hidden matrix is known; the goal is to remove the noise from the observed pixels. Thus these denoising techniques cannot be used to solve the RECONSTRUCT problem.

The problem of reconstructing 0/1 matrices has also been studied recently in data mining. The problem setting considered by Vuokko and Terzi [19] is that a randomized version of the data is revealed and the goal is to reconstruct the original data as accurately as possible. External knowledge about the relationships between the rows and the columns of the noisy observed matrix is also considered by the reconstruction algorithm. The method strongly relies on the fact that the observed matrix is a noisy version of the original one; in fact, in this case even the amount of noise is assumed known. Again, neighborhood information in the form considered in our work cannot be incorporated into the framework proposed in [19].

Matrix decomposition. The analysis of binary data using matrix decompositions has been extensively studied [15], [16], [17]. Although these methods do focus on 0/1 matrices, their goal is to find a low-rank representation of a known input matrix. In some sense, we try to achieve the opposite: We want to extract a hidden 0/1 matrix using some knowledge of its low-rank decompositions, as encoded in the neighborhood data.

Database security. Database privacy questions—in particular the possibility to reconstruct data from seemingly secure, anonymized information—have already been considered in the 70ss [4]. The question is here what information can be inferred about the data, if answers to a small set of statistical questions are known. More recent work by Mielikäinen [14] is concerned with the inverse frequent itemset problem, which aims to infer the contents of a transaction database given an anonymized version of that database and true frequent itemset data.

VI. CONCLUSIONS AND DISCUSSION

There are countless types of real life data that can be described in terms of a bipartite graph. In many cases, the original data is sensitive and cannot not be made public. Thus data owners may consider to publish aggregate information instead. In this paper, we assume that the data owner reveals a particular type of aggregate information, i.e., neighborhood information. The neighborhood information consists of the number of common neighbors between every pair of nodes. Given such information, we asked the following simple question: Can we reconstruct the underlying graph using only this neighborhood information?

We answered this question in an affirmative way by developing a method that reconstructs the biadjacency matrix of the underlying graph. Intuitively, our method reconstructs the components of the matrix' singular value decomposition from the neighborhood information. These components are subsequently used to obtain a (binary) estimate of the hidden matrix. Our experiments suggest that the underlying matrix can be reconstructed with low error or, in some cases, without any error. If, however, the data owner publishes only partial neighborhood information, i.e., hides the degrees of all of the nodes, reconstruction is significantly less effective.

We believe that our work is an instance of a more general problem, in which aggregate characteristics of a dataset are revealed and the question is whether one can infer individual data points. Computational tools that address such types of questions can prove valuable to data owners, who can use them to quantify how much information is leaked by the revealed aggregates.

ACKNOWLEDGMENTS

This research was supported in part by NSF grants CNS-1017529 and IIS-1218437 as well as gifts from Microsoft, Google and Yahoo!. The authors also wish to thank Mark Crovella for valuable discussions during the process.

REFERENCES

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 2011.

[2] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *CVPR*, pages 648–655, 1998.

[3] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[4] F. Y. Chin. Security in statistical databases for queries with small counts. *ACM Trans. Database Syst.*, 3:92–104, 1978.

[5] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW*, pages 271–280, 2007.

[6] P. Drineas, R. Kannan, and M. W. Mahoney. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM J. Comput.*, 36(1):158–183, 2006.

[7] C. Eckhart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, pages 211–218, 1936.

[8] G. H. Golub and C. F. V. Loan. *Matrix Computations*. The John Hopkins University Press, 1996.

[9] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan. Learning influence probabilities in social networks. In *WSDM*, pages 241–250, 2010.

[10] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, 2010.

[11] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):147–159, 2004.

[12] Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *TKDD*, 4(1), 2010.

[13] S. Lattanzi and D. Sivakumar. Affiliation networks. In *STOC*, pages 427–434, 2009.

[14] T. Mielikäinen. Privacy problems with anonymized transaction databases. In *Discovery Science*, pages 219–229, 2004.

[15] P. Miettinen. The boolean column and column-row matrix decompositions. In *ECML/PKDD*, page 17, 2008.

[16] P. Miettinen. Sparse boolean matrix factorizations. In *ICDM*, pages 935–940, 2010.

[17] P. Miettinen and J. Vreeken. Model order selection for boolean matrix factorization. In *KDD*, pages 51–59, 2011.

[18] E. Scheinerman and K. Tucker. Modeling graphs using dot product representations. *Computational Statistics*, 25:1–16, 2010.

[19] N. Vuokko and E. Terzi. Reconstructing randomized social networks. In *SDM*, pages 49–59, 2010.

[20] E. Zheleva and L. Getoor. To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles. In *WWW*, 2009.

[21] E. Zheleva, H. Sharara, and L. Getoor. Co-evolution of social and affiliation networks. In *KDD*, pages 1007–1016, 2009.