Assessing the Significance of Data Mining Results on Graphs with Feature Vectors

Stephan Günnemann • Phuong Dao ○ Mohsen Jamali ○ Martin Ester ○

•RWTH Aachen University, Germany ○Simon Fraser University, Canada

guennemann@cs.rwth-aachen.de {pdao, mohsen_jamali, ester}@cs.sfu.ca

***Abstract*—Assessing the significance of data mining results is an important step in the knowledge discovery process. While results might appear interesting at a first glance, they can often be explained by already known characteristics of the data. Randomization is an established technique for significance testing, and methods to assess data mining results on vector data *or* network data have been proposed. In many applications, however, both sources are *simultaneously* given. Since these sources are rarely independent of each other but highly correlated, naively applying existing randomization methods on each source separately is questionable.**

In this work, we present a method to assess the significance of mining results on graphs with binary features vectors. We propose a novel null model that preserves correlation information between both sources. Our randomization exploits an adaptive Metropolis sampling and interweaves attribute randomization and graph randomization steps. In thorough experiments, we demonstrate the application of our technique. Our results indicate that while simultaneously using both sources is beneficial, often one source of information is dominant for determining the mining results.

I. Introduction

There is an increasing amount of applications generating and handling network data in combination with vector data. While single objects are described by a vector representation, e.g. binary features to describe specific occurrences of characteristics, the relationship between different objects is given by the underlying network structure. Examples for such data include social and rating networks (e.g. Epinions, Flixter), co-author and citation networks (DBLP, Arxiv), gene interaction networks (BioGRID) and sensor networks.

Several data mining algorithms have been proposed to analyze this kind of data: they range from clustering methods [1], [2], [3], [4] over recommendation algorithms [5], [6], [7] to methods analyzing social influence [8], [7]. By combining both data sources, i.e. network and vector information, they try to extract more useful and interesting patterns.

While often the primary focus is the development of novel algorithms, assessing the significance of the mining results has received less attention. In many cases, the question remains open whether the detected result is really interesting. Are the mining results significant or do they just occur due to some already known (and thus uninteresting) characteristics of the data? For example, it is a well-studied phenomenon that individuals with similar characteristics are more likely be connected than arbitrary individuals – the so called homophily [9], [8]. Thus, detecting this pattern is not beneficial for further understanding the data. A consequential question is: do the existing methods really gain a benefit by combining both data sources or do their results just occur due to one type – vector data *or* network data?

In this work, we present a method to assess the significance of data mining results on graphs with binary feature vectors. We use the principle of randomization-based significance testing. That is, we generate multiple random datasets that retain well-known characteristics of the original data encoded by an appropriate null model. If the results on the original data deviate significantly from the results on the random data, then the detected data mining results are assessed as significant.

Recently some randomization approaches have been introduced in the field of data mining [10], [11], [12], [13], [14]; though, they mainly focus on either randomizing vector data *or* network data. A first attempt to handle our setting might be to apply the existing approaches on both sources *independently* (as, e.g., done in [15]). This procedure, however, is highly questionable due to properties as homophily: the two sources are not independent. Intuitively, if we did not consider the dependency between the graph structure and the feature vectors, the randomized data will be 'too random', i.e. important structure is destroyed by the randomization. As a consequence, the data mining result on the original data might appear to be significant even if it is not. Therefore, as our main contribution, we introduce a novel null model for graphs with binary feature vectors that incorporates these insights and preserves characteristics exploiting both sources simultaneously and dependent on each other.

Furthermore, generating the random datasets is highly challenging for our setting: we cannot simply generate one observation (e.g. vertex) after the other as it, for example, can be done in usual vector databases [14]. In the latter case, the sampled observations can be assumed to be (almost) independent, i.e. given an appropriate model of the data, the samples are drawn independently from this model (i.i.d. assumption). Obviously, the independence assumption does not hold in our setting – neither between the vertices nor between the edges. Thus, to cope with the data generation challenge, we introduce a swap randomization technique [10] exploiting Metropolis sampling and interweaving attribute randomization and graph randomization steps.

II. Background and Related Work

Significance testing is frequently used in statistics and typically based either on analytical expressions or on randomization/permutation tests. In this work, we consider randomization tests [16]. For assessing a data mining result on a graph G, a structural measure $M(G)$, e.g. the network modularity, that summarizes the result in a single measure needs to be given. The significance of $M(G)$ on the original graph G is tested as follows: first, we have to specify a set of characteristics of the input graph we want to preserve (or we are already aware of, background knowledge). These characteristics define the null model/hypothesis, and all graphs following this model can be collected in the graph sample space $\mathcal{G}_G$. Second, based on the null model, a set of random graphs $\mathcal{R} \subseteq \mathcal{G}_G$ having similar characteristics as G is obtained by independently drawing graphs from $\mathcal{G}_G$. On each of these graphs $G_i \in \mathcal{R}$ the same data mining algorithm is applied and the structural measure $M(G_i)$ is computed. Finally, the significance of the mining result on the original graph is determined by the (one-tailed) empirical p-value[1]:

$$\frac{|\{G_i \in \mathcal{R} \mid M(G_i) \geq M(G)\}| + 1}{|\mathcal{R}| + 1} \tag{1}$$

If the p-value is sufficiently small, the null hypothesis is rejected, i.e. we can assume that the obtained result is not a consequence of the preserved background knowledge.

Swap randomization techniques: Swap randomization is a special instance of randomization techniques, where local transformations are used to generate the random data. [10] were the first to introduce swap randomizations to access the significance of data mining results. They consider binary matrices while preserving the column and row margins. [11] extended the method for real value matrices.

Swap randomization has been also applied to (unlabeled) graphs under preservation of, e.g., the avg. clustering coefficient [12] or the eigenvalues of the adjacency matrix [13]. The work of [12] additionally aims at preserving the *individual* vertex degrees. Since this might be often too restrictive [12], especially in our case, where each vertex is additionally attached with a feature vector, the authors propose a technique to preserve the degree *distribution*. This method, however, has a severe limitation: if the graph has gaps in the degree distribution, i.e. for certain degree values no vertices exist, then no edges are exchanged between the sets of vertices separated by the gaps [12]. Such gaps, however, frequently occur. In the Epinions data (cf. Sec. V) we observe 82 gaps, leading to 83 disjoint vertex sets between which no edges are exchanged. To avoid this limitation, we present a novel technique that approximately but accurately preserves the degree distribution.

Most importantly, none of the above methods considers attribute data in combination with graph data.

In the field of privacy-preserving graph mining, the work of [15] considers randomization of networks where vertices are annotated with feature vectors: their aim is to reconstruct the original data given the randomized counterparts. To generate the random data, however, they simply consider each source independently. In [17] multi-relational data is considered. However, they do not propose a way to randomize all sources simultaneously; instead, again each relation is randomized independently. Overall, none of the existing works considers the simultaneous randomization of graph and attribute data while preserving characteristics resulting from the inherent dependency between these two sources.

Alternative randomization techniques: An alternative to swap randomization are closed form models, e.g., based on the principle of Maximum Entropy [14]. In contrast to performing local transformations, they generate the random data from scratch by (independently) drawing samples (e.g. edges) from a given model. For example, the well-known Erdös-Rényi graph model corresponds to the Maximum Entropy Model preserving the number of vertices and edges in an unlabeled graph. A single random graph is constructed by independently drawing the edges according to this model.

While the principle of Maximum Entropy was successfully applied to binary and real valued attribute data [14], its (direct) application in our setting is questionable: First, we have to draw samples of two different kinds – edges as well as vertex labels. And second, these samples are rarely independent from each other.

III. A Novel Null Model

In the following, we describe our null model. We start by introducing the characteristics we want to preserve, followed by proposing a principle how to ensure that the randomized graphs accurately (but not necessarily exactly) reflect these characteristics. Finally, the overall model and an extension to handle unknown values are given. Before presenting details, we first formalize the considered data type:

Definition 1: [Basic notions] A graph $G = (V, E, l)$ with binary feature vectors is defined by a set of vertices V, a set of edges $E \subseteq V \times V$, and a labeling function $l : V \rightarrow \{0,1\}^D$. $D =: D(G)$ is the dimensionality of the feature vectors. We denote with $l(v, d)$ the value of vertex v in the d-th dimension. The set of all adjacent vertices w.r.t. $v \in V$ is given by $N(v) = \{u \in V \mid (v, u) \in E\}$. For a given graph G, we denote the set of vertices/edges by $V(G)/E(G)$.

Characteristics to Preserve. Besides the number of vertices, the number of edges, and the dimensionality of the data, [10], [12] introduced further characteristics considering only one type of data. These characteristics include the row and column sums as well as the vertex degrees. Formally,

- row sum: $\mathbf{1}_G(v) = |\{d \in D \mid l(v, d) = 1\}|$
- column sum: $\mathbf{1}_G(d) = |\{v \in V \mid l(v, d) = 1\}|$
- vertex degree: $\delta(v) = |N(v)|$

Even though these characteristics consider only a single type of information, we still aim at preserving them; otherwise the mining results can hardly be compared. However, the

[1]Here we assumed that the larger the value of $M(G)$, the stronger the pattern in the data (e.g. modularity). Otherwise (e.g. graph-cut values should be small) we can simply reverse the inequality.

graph and feature data are rarely independent. Thus, in the following we introduce a characteristic simultaneously considering both data sources.

Average neighborhood correlation: In network data it is known that the attribute values of vertex v dependent on the values of its adjacent vertices $N(v)$. This effect is known as homophily or social correlation and has been studied extensively [9], [8]. The average correlation between the attribute values of adjacent vertices, e.g., is significantly higher than the correlation between two random vertices. Thus, a first idea would be to consider the average correlation between the feature vectors of any pair of adjacent vertices. Since, however, the social correlation might vary greatly throughout the network, i.e. some vertices show high correlation while others show much lower values, the overall average is a too rough characteristic. Thus, we propose to consider for *each* vertex the average correlation to its neighboring vertices. A real world example, for the distribution of the average correlation values is shown in the experimental section (Fig. 2): For the DBLP data, around 40% of all vertices have an avg. correlation close to 1, while almost no vertices have a negative avg. correlation (details are discussed later).

More precisely, we propose to use the (sample) Pearson correlation coefficient between vertex u and v, defined by

$$r(u,v) = \frac{\sum_{d=1}^{D}(l(u,d)-\tilde{l}(u))(l(v,d)-\tilde{l}(v))}{\sqrt{\sum_{d=1}^{D}(l(u,d)-\tilde{l}(u))}\sqrt{\sum_{d=1}^{D}(l(v,d)-\tilde{l}(v))}}$$

where $\tilde{l}(.)$ denotes the avg. attribute value of the corresponding vertex. However, we have to take special care to obtain our final characteristic: Since correlation is not an additive measure, we should not simply average over the obtained correlation values. As shown in [18], the variance of r grows smaller as the correlations gets closer to 1. Although many works are not aware of this issue, simply using the (arithmetic) mean of correlation values produces misleading results. Instead, we employ the Fisher transformation to obtain an additive measure – so-called Fisher z values. After the Fisher transformation, the variance of the correlation r is approximately constant for all values of the sample correlation coefficient. The Fisher transformation z and its inverse z^{-1} are given by

$$z(r) = \frac{1}{2}\ln\frac{1+r}{1-r} \qquad z^{-1}(z) = \frac{e^{2z}-1}{e^{2z}+1}$$

Please note that for practical applications, correlation values of exactly ± 1 need to be replaced with $\pm(1-\epsilon)$ for some arbitrarily small ϵ. Considering these aspects, the average correlation for a vertex v to its neighboring vertices can now be computed based on the following equation:

$$\varnothing Corr_G(v) := z^{-1}\Big(\frac{1}{|N(v)|}\sum_{u\in N(v)} z\big(r(u,v)\big)\Big) \qquad (2)$$

This characteristic represents in a well-founded way the dependency between network and feature information.

Accurate Preservation of Characteristics. Our goal is to define the set of random graphs having a similar avg. correlation distribution and a similar degree distribution as the input graph G. For this purpose, we perform hypothesis testing to assess whether the characteristic distributions of the graphs originate from the same distribution. We require *similar* distributions since (a) for the continous-valued correlation values we cannot expect to find two graphs showing *exactly* the same characteristics, and (b) the random graphs should generalize the structure, i.e. it is not suitable to require that each vertex *individually* shows exactly the same correlation (or degree). We allow small deviations between the distributions without loosing accuracy.

The distribution of the average correlation values is given by the following empirical distribution function

$$F^G_{corr}(x) = \frac{1}{|V|}\cdot|\{v\in V \mid \varnothing Corr_G(v)\leq x\}|$$

which represents the fraction of samples with average correlation values less than or equal to x. Accordingly, we define the function $F^G_{deg}(x)$ that represents the degree distribution.

To assess the goodness of fit between two distributions F^G and $F^{G'}$ originated from two graphs G and G', we use a 'two sample test'. That is, we test the hypothesis that the two samples (e.g. the two observed sets of correlations) come from the same (unspecified) distribution. We denote this null hypothesis with $F^G \sim F^{G'}$. For our purpose, a 'two sample test' is highly advantageous compared to the frequently used 'one sample tests'. While in 'one sample tests', we have to assume a specific parametric family for the distributions, in 'two sample tests', we do not have to make any assumptions about the actual shape of the distributions; the distribution remains unspecified. Thereby we can handle any distribution including the frequently observed Normal and Power-low distributions.

More precisely, we use the two sample Cramer-von-Mises test [19], to test the hypothesis $F^G \sim F^{G'}$. The Cramer-von-Mises criterion is defined by the random variable T

$$T = \frac{|V|\cdot|V'|}{|V|+|V'|}\int_{-\infty}^{\infty}[F^G(x)-F^{G'}(x)]^2\ \mathrm{d}H^{G,G'}(x)$$

where F^G, $F^{G'}$ are two distribution functions of graphs $G=(V,E,l)$, $G'=(V',E',l')$ and $H^{G,G'}$ is given by $H^{G,G'}(x) = \frac{|V|\cdot F^G(x)+|V'|\cdot F_{G'}(x)}{|V|+|V'|}$, which can be seen as a kind of average distribution. Intuitively, the random variable T 'measures' the difference between two distributions. Now, let T_{obs} be the observed difference between two given (empirical) distribution functions. If the probability $P(T\geq T_{obs}|F^G\sim F^{G'})$ is smaller than a significance threshold α, the null hypothesis is rejected[2]. In this case, the probability to observe the difference T_{obs} (or an even

[2]Note that the value of T_{obs} can be efficiently determined based on a ranking of the observations [19] and the p-value $P(T\geq T_{obs}|...)$ can also be computed analytically.

extremer one) is very small under the assumption that the samples come from the same distribution. Thus, the two distributions differ significantly.

Overall model. Based on the above discussion, we define the set of random graphs that have similar properties as the input graph by the following sample space:

Definition 2: [Graph sample space $\mathcal{G}_G$]
Given an input graph $G = (V, E, l)$, the sample space $\mathcal{G}_G$ contains all graphs $G' = (V', E', l')$ with

- *$V = V'$, i.e. the set of vertices is identical,*
- *$|E| = |E'|$ and $D(G) = D(G')$, i.e. the number of edges and the dimensionality of the feature vectors is preserved,*
- *$\forall v \in V : \mathbf{1}_G(v) = \mathbf{1}_{G'}(v)$ and*
  $\forall d \in \{1, \ldots, D(G)\} : \mathbf{1}_G(d) = \mathbf{1}_{G'}(d)$
  the row and column sums are preserved,
- *$\neg\big[P(T \geq T_{corr} \mid F^{G'}_{corr} \sim F^{G}_{corr}) < \alpha_{corr}\big]$*
  the distribution of the correlation values is preserved,
- *$\neg\big[P(T \geq T_{deg} \mid F^{G'}_{deg} \sim F^{G}_{deg}) < \alpha_{deg}\big]$*
  the distribution of the vertex degrees is preserved.

Please note that we have to require $\neg[P(...) < \alpha]$ since we want to ensure similar distributions. We do *not* want to get a significant difference. Furthermore, the row and column sums are preserved for each vertex and dimension *individually* due to three reasons: First, this property can be easily preserved exactly [10]. Second, for each of the above defined graphs we can find a graph with an identical row/column sum *distribution* by simply permuting the numbering of vertices/dimensions. Our definition avoids many isomorphic graphs that just differ in their numbering of vertices/dimensions. For each graph $|V|! \cdot |D(G)|!$ many graphs with a permuted numbering exist, which can safely be removed from our considerations avoiding an artificial blow up of the graph sample space. Finally, preserving the row sum for each vertex corresponds to preserving their avg. attribute values. This is especially useful for social rating networks, where users tend to give ratings in a specific range. Thus, the 'user bias' [20] is preserved. The same argument applies for the column sums ('item bias').

Extension to Unknown/Missing Values. In many real world applications we do not observe 'perfect' binary features but unknown or missing values frequently occur. In social rating networks, for example, users usually do not rate all products but just subsets of them. Missing values occur due to measurement errors. Our null model can be extended to handle unknown and missing values (denoted with the symbol $\perp$). First, we additionally require to preserve for each vertex and dimension the number of unknown values:

- *$\forall v \in V : \perp_G(v) = \perp_{G'}(v)$ and*
  $\forall d \in \{1, \ldots, D(G)\} : \perp_G(d) = \perp_{G'}(d)$
  the number of unkown values per vertex and dimension is preserved,

where $\perp_G(x)$ denotes the number of unknown values per vertex/dimension x. Second, we have to adapt the computation of the correlation values. The correlation between vertices u and v must only be computed based on those dimension which are defined for both vertices, i.e. based on

$$d(u,v) = \{d \in D \mid l(u,d) \neq \perp \wedge l(v,d) \neq \perp\}$$

Since now the correlation values between a vertex v and its neighbors are based on different sample sizes $|d(u,v)|$, also the average correlation value has to be adapted. Therefore, we use a method for synthesizing correlation matrices [21], where we weight the different correlations according to their (reciprocal) variance. By using the Fisher transformation, the variance of $z(r(u,v)))$ is simply given by $\sigma^2 = \frac{1}{n-3}$ where n is the sample size. Thus, the adapted avg. correlation is

$$\varnothing Corr_G(v) := z^{-1}\Big(\frac{1}{|N(v)|}\sum_{\substack{u \in N(v)\\ |d(u,v)|>3}} |d(u,v) - 3| \cdot z\big(r(u,v)\big)\Big)$$

IV. Randomization Technique

In the following, we describe how to generate a set of random graphs according to our null model. Our goal is to uniformly sample graphs from the probability density function $\pi(G_i) = \{\frac{1}{|\mathcal{G}_G|}, \text{ if } G_i \in \mathcal{G}_G;\ 0, \text{else}\}$. This function is well-defined since the graph sample space is finite. Since direct sampling from π is difficult, we use an (adaptive) Metropolis-Hastings approach [22], [16]. That is, starting with the input graph, we use local transformations (i.e. swaps) to generate a sequence of modified graphs. An overview of our algorithm is presented in Algorithm 1.

Formally, the Metropolis-Hasting approach is based on a Markov chain and operates as follows: Instead of directly drawing samples from the distribution π, we iteratively draw samples according to a (more simple) proposal density $q(G, G')$ that depends on the current state G of a Markov chain. In our case, each state corresponds to a graph G. Starting with the input graph G as the current state of the chain, we randomly generate a graph G' based on $q(G, G')$. With probability $\alpha(G, G') = \min\{\frac{\pi(G') \cdot q(G', G)}{\pi(G) \cdot q(G, G')}, 1\}$, the graph G' is 'accepted' and selected as the novel state of the chain. Otherwise, the sample G' is 'rejected', and we keep staying at the old state G. The procedure is repeated S times, where S is set so that the chain has mixed (Corr. 1).

The crucial part is defining a meaningful proposal density q. In our case, we aim at interweaving attribute and graph randomization steps. For this, we have to take care that both sources are randomized evenly; i.e., we have to ensure that not just only one source is randomized sufficiently, while the other source still remains almost unchanged. Therefore, we propose to use an interweaving probability I that controls the fraction of graph vs. attribute randomization (we describe the derivation of I later on). Based on the interweaving probability we either perform an *edge swap* (lines 6-12) or an *attribute swap* (lines 13-19).

Edge swaps randomize the graph as illustrated in Fig. 1 (left): we randomly select an edge $(u,v) \in E(G)$ and a

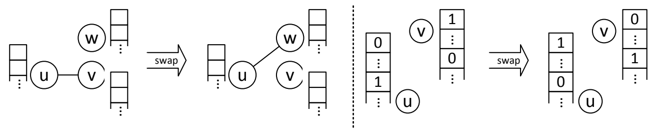

Figure 1. Edge swap (left) and attribute swap (right)

vertex $w \in V(G)$, and relocate the endpoint of the edge to w. Note that some of these swaps might be invalid, e.g. if the edge (u, w) already existed in G, we would get a multi-edge in G', which is not permitted in our data. These invalid swaps are handled as self-loops in the Markov chain, i.e. they point to the current state G. We denote with $T_e(G)$ all graphs reachable from G by a single, valid edge swap. In contrast to [12], our proposed swap handles gaps in the degree distribution (cf. Sec. II). We do not enforce to get exactly the same degree distribution but we guarantee similar ones due to the overall processing (cf. line 11).

An attribute swap is given in Fig. 1 (right): We randomly select two vertices u and v as well as two dimensions d_1 and d_2, and for each selected vertex we interchange the attribute values in dimension d_1 and d_2 (similar to [10]). Again, some of these swaps might be invalid since they would, e.g., not preserve the column sums. We denote with $T_a(G)$ all graphs that can be reached from G by a single, valid attribute swap. Note that $T_e(G) \cap T_a(G) = \emptyset$ holds.

Since we used self-loops for the invalid swaps, the degree of each state in the Markov chain is identical, i.e. $|E| \cdot |V| + |V|^2 \cdot |D|^2$. Accordingly, we ensure a uniform sampling from π by first deciding (based on I) which type of swap to use and then uniformly performing one (valid or invalid) edge/attribute swap. Thus, the proposal density we use is

$$q(G, G') = \begin{cases} I \cdot \frac{1}{|E|\cdot|V|} & G' \in T_e(G) \\ (1-I) \cdot \frac{1}{|V|^2 \cdot |D|^2} & G' \in T_a(G) \\ 1 - I \cdot \frac{|T_e(G)|}{|E|\cdot|V|} - (1-I) \cdot \frac{|T_a(G)|}{|V|^2\cdot|D|^2} & G' = G \\ 0 & else \end{cases}$$

Finally, having sampled a graph G' according to q, we have to evaluate the acceptance probability $\alpha(G, G')$. In our case, the probability simplifies to 1 if $G' \in \mathcal{G}_{G_{in}}$ and 0 otherwise. (Note: $G \in \mathcal{G}_{G_{in}}$ holds by induction.) Thus, for accepting graph G' as the novel state, the property $G' \in \mathcal{G}_{G_{in}}$ is checked in lines 11 and 18. This check can be done efficiently since (based on G) the characteristics of just a few vertices have to be updated in G': After an edge swap, the avg. correlation values for $\{u, v, w\}$ have to be recomputed since their neighborhood has changed. After an attribute swap, the vertices $\{u, v\} \cup N(u) \cup N(v)$ have to be updated since by swapping the attribute values of $\{u, v\}$ also the neighbors are affected. Overall, if G' is accepted, it is ensured that all characteristics specified in Def. 2 are preserved.

Interweaving probability. An important question is how to set the interweaving probability I. As we will see in the experiments, the intuitive choice 0.5 does not lead to a balanced randomization of both sources. To derive I, we consider three issues: 1) How many edge/attribute swaps per source should be performed to get a most dissimilar graph? 2) What is the probability to perform an invalid swap (lines 8,15)? 3) What is the probability to accept a valid swap (lines 11,18)?

```
    Input: Graph $G_{in}$ with binary feature vectors
    Output: Randomized graph $G$
1   initialize $\Delta_e, \Delta_a, \gamma_e, \gamma_a$ according to Eq. 3 & 4, Theo. 1 & 2
2   $\beta_a = 0.5, \beta_e = 0.5, s = 0$
3   $G \leftarrow G_{in}$                                   // initial state of chain
4   while $s < S$ do
5       with probability $I$ (Eq. 5) do $swap \leftarrow e$ else $swap \leftarrow a$
6       if swap=e then                                      // edge swap
7           select edge $(u, v) \in E(G)$, select vertex $w \in V(G)$
8           if $w \neq u \wedge w \neq v \wedge (u, w) \notin E(G)$ then
9               $G' \leftarrow G$
10              $E(G') \leftarrow (E(G_r) \backslash \{(u, v)\}) \cup \{(u, w)\}$
11              if $G' \in \mathcal{G}_{G_{in}}$ then $G \leftarrow G'$     // characteristics preserved?
12              adapt $\beta_e$
13      else                                                //attribute swap
14          select $u, v \in V(G)$, $d_1, d_2 \in D(G)$
15          if
            $l(u, d_1) = l(v, d_2) \wedge l(u, d_2) = l(v, d_1) \wedge l(u, d_1) \neq l(u, d_2)$
            then
16              $G' \leftarrow G$
17              $l'(x, d_1) \leftarrow l(x, d_2), l'(x, d_2) \leftarrow l(x, d_1), x \in \{u, v\}$
18              if $G' \in \mathcal{G}_{G_{in}}$ then $G \leftarrow G'$     // characteristics preserved?
19              adapt $\beta_a$
20      $s$++
21  return $G$
```

Algorithm 1: Adaptive swap randomization for graphs with binary feature vectors

Issue 1: Given two graphs G and G', they are most dissimilar w.r.t. their edges if $E(G) \cap E(G') = \emptyset$ holds. However, if the graphs are very dense, the edge intersection might not become empty. Thus, highest dissimilarity is obtained by minimizing the intersection. Starting with G, a graph G' with minimal intersection can be obtained by performing

$$\Delta_e := \min\{|E(G)|, 0.5 \cdot (|V|^2 - |V|) - |E(G)|\} \tag{3}$$

edge swaps. If the graph is sparse, we have to relocate $|E(G)|$ edges. If the graph is dense, we swap as many "missing edges" as possible with existing edges. Note that this estimate is optimistic, i.e. it is a lower bound, since we do not consider swaps relocating edges back to their old positions.

A similar result can be obtained for the attribute data. Here we have to take into account the three different labels. We aim at minimizing $|1(G) \cap 1(G')| + |0(G) \cap 0(G')| + |\bot(G) \cap \bot(G')|$, where $x(G) = \{(v, d) \,|\, l(v, d) = x\}$. Starting with G, a graph G' with minimal intersection can be obtained by performing

$$\Delta_a := \frac{1}{4}(A - \max\{2\cdot|1(G)| - A, 2\cdot|0(G)| - A, 2\cdot|\bot(G)| - A, 0\}) \tag{4}$$

attribute swaps, where $A = |V| \cdot |D|$. If the most frequently occurring label occurs more often than the sum of the cardinalities of the two remaining labels, the intersection cannot become empty. This mandatory intersection is subtracted from the overall number of elements. The remaining elements are swapped, where each swap relocates 4 attribute values. This is again a lower bound for the number of swaps.

Obviously, Δ_e and Δ_a may differ. While one source requires only a few swaps to become very dissimilar, the second one needs many swaps. Since both sources should be randomized evenly, the ratio of (valid and accepted) edge swaps w.r.t. the (valid and accepted) attribute swaps should be $\frac{\Delta_e}{\Delta_e+\Delta_a}$.

Issue 2: Not all swaps tried are valid (lines 8,15). This issue has to be incorporated in the interweaving probability I. For the edge and attribute randomization we get

Theorem 1: Given a random edge swap $(u,v) \in E$ and $w \in V$. The probability that the swap is valid is

$$\gamma_e := \left(1-\frac{2}{|V|}\right)\cdot\left(\frac{|V|-1-ED}{|V|-2}\right) = \frac{|V|-1-ED}{|V|}$$

where $ED = \frac{1}{2|E|}\sum_{a\in V}\delta(a)^2$ is the expected degree of an incident vertex when selecting a random edge.

Proof: With prob. $1-\frac{2}{|V|}$, the vertex w is not equal to u and v. Given an edge (u,v) and the degree $\delta(u)$ of node u. The probability that w is not adjacent to u is given by $1-\frac{\delta(u)-1}{|V|-2} = \frac{|V|-1-\delta(u)}{|V|-2}$ (since u is connected to v, its degree is decreased by 1). Finally, since the edges are uniformly drawn from E, the expected degree of u is given by $ED = \frac{1}{|E|}\sum_{(a,b)\in E}\frac{\delta(a)+\delta(b)}{2} = \frac{1}{2|E|}\sum_{a\in V}\delta(a)^2$. Since vertices with a higher degree have a higher probability to become selected, we get the squared dependency w.r.t. δ. ■

Theorem 2: Given a random attribute swap $u,v \in V$ and $d_1,d_2 \in D$. The probability that the swap is valid is estimated by $\gamma_a := 2\cdot(p_{\mu_1}^2 p_{\mu_0}^2 + p_{\mu_1}^2 p_{\mu_\perp}^2 + p_{\mu_0}^2 p_{\mu_\perp}^2)$ where $p_{\mu_1} = |1(G)|/(|V|\cdot|D|)$ and p_{μ_0}, $p_{\mu_\perp}$ accordingly.

Proof: The equation can simply be obtained by assuming that the attribute values are uniformly distributed over the attribute matrix. ■

Issue 3: Besides invalid swaps, a swap may be rejected since it does not follow the null model, e.g. its avg. correlation distribution is significantly different. We denote the probability that a valid edge/attribute swap is accepted with β_e/β_a. Estimating these probabilities is highly complex since they do not only depend on the number of vertices involved in the swap but also on the actual distribution of the attribute values. Thus, we refer to a different principle: We *learn* the acceptance probabilities β_e, β_a during the run of the algorithm. That is, our overall algorithm is an instance of an internal adaptive Markov Chain Monte Carlo method [22]. This is indicated in lines 12 and 18 of the algorithm.

More precisely, instead of updating the parameters continuously, we use a more efficient batch update at time points $\{T_j \mid j > 0\}$, which are measured w.r.t. to the valid swaps. The update equation is given by

$$\beta_* \leftarrow \frac{1}{2}\cdot\beta_* + \frac{1}{2}(T_{j+1}-T_j)^{-1}\cdot AS_j^* \qquad * \in \{e,a\}$$

where AS_j^* denotes the number of accepted edge/attribute swaps since the last parameter update at time T_j. We use an exponentially increasing update time, i.e. $T_j = 2^j$; thus ensuring $\lim_{j\to\infty} T_{j+1}-T_j = \infty$ and the values of β_* converge. Learning the probabilities β_e and β_a allows us to adapt to the characteristics of the given data set.

Overall, we now estimate the interweaving probability I: Assume we are trying x swaps. The number of valid and accepted edge swaps is given by $x\cdot I\cdot\gamma_e\cdot\beta_e =: y_e$. Accordingly, the number of valid and accepted attribute swaps is given by $x\cdot(1-I)\cdot\gamma_a\cdot\beta_a =: y_a$. Based on issue 1, it should hold $\frac{y_e}{y_e+y_a} = \frac{\Delta_e}{\Delta_e+\Delta_a}$ to obtain a balanced randomization of both sources. Solving these equations to I yields

$$I = \frac{\Delta_e\cdot\gamma_a\cdot\beta_a}{\Delta_e\cdot\gamma_a\cdot\beta_a + \Delta_a\cdot\gamma_e\cdot\beta_e} \qquad (5)$$

Estimating the number of swaps. In the following we analyze the convergence/mixing time of the Markov chain. The number of performed swaps in Algo. 1 is guided by the parameter S (line 4). We provide a recommendation for the value of S. Proving the convergence of a Markov chain is highly complex and an open research question, so our estimate provides only a lower bound on the number of swaps needed to perform to make the chain mixed. Though, as we will show in the experiments, our result indeed yields a good estimation for the required number of swaps. Additionally, even if we cannot guarantee that the chain has converged, we can expect conservative, i.e. larger, p-values.

We exploit the results presented above. Let us focus on the randomness w.r.t. the edges (the derivation for the attributes is identical). W.l.o.g. we assume $\Delta_e = |E(G)|$ (i.e. the graph is sparse; otherwise we can swap the 'non-edges'). As discussed, to obtain a most dissimilar graph, Δ_e many edges need to be relocated. Based on the previous results, trying S swaps leads to an estimated number $S_e := S\cdot I\cdot\gamma_e\cdot\beta_e$ of valid and accepted edge swaps. Since we do not prohibit to swap an edge several times, i.e. an already relocated edge can again be relocated, we do not necessarily swap all different edges from Δ_e. In the worst case, for example, the S_e swaps relocate the same edge again and again. However, we can estimate the probability that all edges from Δ_e are swapped at least once.

Theorem 3: Given the number S_e of valid and accepted edge swaps. The probability P_e that each of the Δ_e many edges is swapped at least once can be approximated by

$$\log P_e \approx \sum_{i=0}^{\Delta_e-1}\log(S_e-i) - \Delta_e\cdot\log(\Delta_e) + h(S_e,\Delta_e)$$

where $h(n,m) = \frac{1}{2}(\log(1-\frac{m}{n}) - \log(-\frac{n}{m}+x_0+1)) + (n-m)\log(\frac{n}{m}-1) + m\log(e^{x_0}-1) + m - n\log(x_0) - n$ and x_0 is the positive solution of $\frac{m}{n}x = 1-e^{-x}$

Proof: (Sketch) We can reduce our problem to the problem of counting the number of *injective* functions defined on a domain Dom covering S_e elements (each element corresponds to a single swap x) and an image Im covering Δ_e elements (which edge is swapped by x). By considering

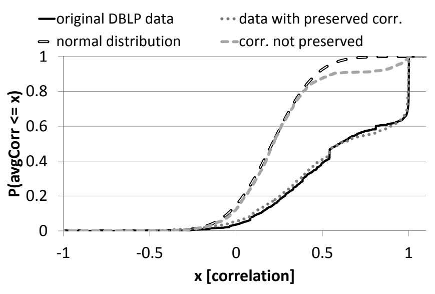

Figure 2. Plot of F^G_{corr}, i.e. fraction of vertices having an avg. correlation *smaller* than x.

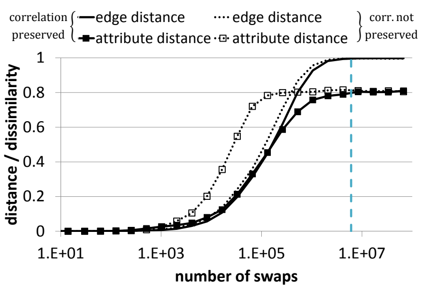

Figure 3. Convergence on DBLP data; vertical line indicates estimated number of swaps

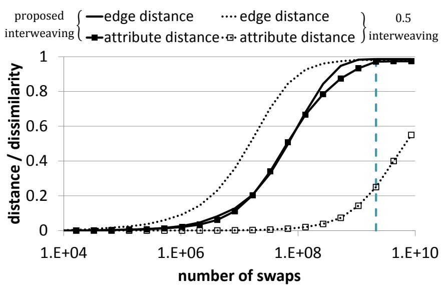

Figure 4. Convergence on Arxiv data; vertical line indicates estimated number of swaps

injective functions, each element from Im is at least used once. Dividing by the number of all possible functions, i.e. $(\Delta_e)^{S_e}$, leads to the desired probability.

The number of injective functions is known to be $\Delta_e! \cdot \left\{ \begin{smallmatrix} S_e \\ \Delta_e \end{smallmatrix} \right\}$ where $\left\{ \begin{smallmatrix} n \\ m \end{smallmatrix} \right\}$ denotes the Stirling number of the second kind. Stirling numbers of the second kind can be approximated by $m^{n-m} \cdot \binom{n}{m} \cdot e^A \cdot f(t_0)$ where A, f and t_0 are given in [23]. Using this approximation, after reformulation and simplification we obtain the probability P_e. ■

The same derivation can be done for the attribute swaps, leading to the probability P_a when S_a attribute swaps were valid and accepted. Based on these probabilities, we use the principle of significance testing, to estimate the number of recommended swaps: If the probabilities $1 - P_e$ and $1 - P_a$ are small enough, the hypothesis that the randomized graph is *not* sufficiently different can be rejected.

Corollary 1: The number of swaps S to obtain a sufficiently randomized graph (w.r.t. significance level α_{swap}) should fulfill $\max\{1 - P_e, 1 - P_a\} \leq \alpha_{swap}$ *, where P_e as given in Theo. 3 and $S_e = S \cdot I \cdot \gamma_e \cdot \beta_e$ (and P_a/S_a accordingly)*

Further aspects. An advantage of our algorithm is the potential to easily randomize just one source – edges or attributes. E.g., by artificially setting $\Delta_e = 0$ we do not randomize the edges but only the attributes. Please note that even if only one source is randomized, we still have to check whether the avg. correlation distribution is similar since this distribution depends on both sources. We will demonstrate the effects of randomizing only one source in the experiments. Similar to [11], for some artificial cases, the swaps cannot reach all states of $\mathcal{G}_{G_{in}}$. In this case, however, we can guarantee to get conservative p-values. Finally, forward-backward sampling [16] is used to resolve the dependency to the input.

V. Experimental Analysis

In this section we present the empirical evaluation of our method and highlight the most interesting findings. As common, all significance levels are set to 0.05. We applied our randomization method and significance testing on different data mining methods such as clustering, recommendation and weighted PageRank. All these methods are designed to handle both sources simultaneously.

Datasets. The first dataset is a co-author graph extracted from DBLP[3]. Each vertex represents an author, edges describe co-authorship. Binary features vectors are used to indicate the conferences an author has published at. The top-11 conferences are selected based on their numbers of papers. The graph contains 2482 vertices and 7302 edges. The second data is a citation graph extracted from Arxiv[4]. Each vertex represents a publication, edges correspond to citations. Binary features indicate the keywords used in the papers' abstracts. We used the 300 most frequently occurring keywords. The graph contains 13k vertices and 120k edges. Our third data is a trust/rating network extracted from Epinions[5]. Vertices correspond to users, edges represent trust relationships, and dimensions are items the user can rate. Binary features were obtained by discretizing the integer valued ratings; values smaller than 3 are mapped to 0. Unknown values are present if no rating was performed by a user for a specific item. We used the 500 most rated items. Cold starts users with less than 3 ratings are removed since for these users no meaningful correlations can be computed. The graph contains 9k vertices and 154k edges.

We also created altered versions of these data sets, where only one source is kept, the other source is random. E.g., DBLP$_{attr}$ contains only the attribute information of DBLP; the edges form a random graph (with similar degree distribution as the original data). In these data sets, the correlation between the two sources is already lost. We use these data sets to show the importance of the different data types.

Model analysis. First, we analyze whether our model accurately preserves the correlation distribution. In Fig. 2 the function F^G_{corr} for the DBLP data is shown. A large number of vertices shows a very high correlation confirming the need to preserve this information. Our method successfully preserves this correlation in the randomized graphs; the function F^G_{corr} for the randomized graphs is very similar (in the plot, one instance is illustrated). When generating graphs that do not preserve the correlation, i.e. by setting $\alpha_{corr} = 0$, we get a strongly deviating distribution. Interestingly, in this case the distribution tends to become a normal distribution.

[3]http://dblp.uni-trier.de
[4]http://www.cs.cornell.edu/projects/kddcup/datasets.html
[5]http://www.trustlet.org/wiki/Epinions_dataset

Incorporating the correlation distribution in our null model allows us to keep the dependency between the mere graph structure and the attribute information.

Next, we analyze convergence properties. In Fig. 3, we measure the randomness of the generated graphs w.r.t. the number of performed swaps on the DBLP data. As measure for randomness/dissimilarity we use the Jaccard distance; first, based on the graphs' edge sets and second, based on the positions of 1 entries. We again test a randomization with and without preserving the correlation. In both cases, the dissimilarity to the input graph converges to the same value. Not surprisingly, if the correlation is not preserved, the convergence is obtained faster. More importantly, even when preserving the correlation, we can reach a high dissimilarity. For this data set, the attribute dissimilarity just reaches a value of 0.8 since most of the features values are 1. Fig. 3 also presents the number of recommended swaps as based on Theorem 1 (for the data where the corr. is preserved). We observe that the estimate is really close to the point were the maximal dissimilarity is obtained. The absolute time to reach this point was around ten minutes.

In Fig. 4 we analyze the importance[6] of the interweaving probability I. We measure the randomness, this time on the Arxiv data. In one case, we use the interweaving prob. as proposed by our method, in the other case, we use the choice 0.5. As shown, with our method both sources are randomized almost evenly (as also in Fig. 3). The choice 0.5 leads to a faster increase of the edge randomness; however, at the price of almost not randomizing the attributes. Note the logarithmic scale on the x-axis: the convergence w.r.t. the attribute randomness is several orders of magnitude slower. Our method leads to a fast increase in the randomness of both sources, and, again, our estimate for the number of swaps to perform is close to the point where maximal dissimilarity is obtained.

Assessing Clustering Results. For the clustering task we choose Co-clustering [2] and the method of Shiga et. al. [1], as representatives of distance-based and spectral methods. Four different structural measures are selected to evaluate the results: weighted normalized cut (*w. NCut*), where edge weights are based on the reciprocal L2 distance between the corresp. adjacent vertices; (unweighted) normalized cut (*NCut*); modularity; and the within-cluster sum of squares based on L2 distance (*TD*, total distance). The number of clusters is set to 10. Additional to the clustering methods, we computed the average L1 distance between any pair of adjacent vertices. When preserving the correlation distribution, we expect that this characteristic is not significant. In each experiment, we generated 100 randomized graphs.

[6]Note: In theory, the interweaving prob. can be set arbitrarily; in any case, our method ensures a uniform sampling from the graph sample space when performing *infinitely* many swaps. Though, in practice the number of performed swaps is limited. Thus, obtaining high randomness with only a few swaps is crucial and realized by the proposed interweaving.

	Co-clustering				*Shiga et. al.*				*avg.*
dataset	*w. NCut*	*NCut*	*TD*	*Modul.*	*w. NCut*	*NCut*	*TD*	*Modul.*	*L1*
Random	0.76	0.77	0.52	0.82	0.83	0.83	0.23	0.89	0.47
DBLP	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	1.00
DBLP$_{attr}$	0.40	0.45	**0.03**	0.27	**0.03**	**0.04**	0.55	**0.04**	0.99
DBLP$_{graph}$	0.60	0.59	0.37	0.89	**0.01**	**0.01**	0.82	**0.01**	0.80
Arxiv	**0.01**	**0.01**	0.31	**0.01**	–	–	–	–	0.06
Arxiv$_{attr}$	0.78	0.78	**0.01**	0.97	–	–	–	–	0.41
Arxiv$_{graph}$	**0.01**	**0.01**	**0.01**	**0.01**	–	–	–	–	0.84

Figure 5. P-values for different structural measures by using default randomization (both sources randomized, correlation distrib. preserved)

In Fig. 5 we analyze the results on different data sets (Shiga was not applicable on the larger Arxiv data.) Both sources are randomized and the correlation distribution is preserved. This setting is our default randomization technique. As expected, the results on random data are insignificant for all structural measures. Also, the average L1 is not significant for any data when preserving the correlation.

On DBLP all clustering measures are significant. Considering DBLP$_{attr}$, i.e. the data were we just use the attribute information of DBLP, Co-Clustering only obtains a significant result for TD, which can be explained since the graph is random. However, even on DBLP$_{graph}$, the graph measures are still not significant. This suggests that Co-clustering primarily uses the attributes for clustering, and if this information is not given, the results are not significant. In contrast, Shiga shows insignificant results only for the TD value. One advantage of Shiga is its automatic balancing of the two sources: the source with better clustering structure is used more strongly [1], while Co-Clustering uses both sources even if they show no good grouping.

On the Arxiv data, Co-Clustering yields an insignificant TD value indicating that for Arxiv the attribute values are already mostly described by preserving the neighborhood correlation. Thus, combining the graph with the correlation information is the most valuable source to use for clustering. This is confirmed by the results on the altered data sets: On Arxiv$_{attr}$ the TD value is significant, i.e. the attributes on their own show a good clustering structure, they only become insignificant if the correlation to the graph is taken into account (note: on Arxiv$_{attr}$ the correlation between the sources is already destroyed for the input). On Arxix$_{graph}$, in contrast, all results are significant.

Overall, the results suggest that one source of the data is often preferred or more valuable for the algorithms. In most cases, however, by combining edges and attributes the results are more significant.

In Fig. 6 we analyze the effect when *not* preserving the correlation distribution[7], i.e. setting $\alpha_{corr} = 0$. As shown, the avg. L1 value as well as the TD on the Arxiv data become significant. This confirms the observation made above

[7]This setting is different to considering, e.g., Arxiv$_{attr}$. In Arxiv$_{attr}$, the correlation between the sources is already lost for the input data. Now, the input shows a meaningful correlation, which, however, is destroyed for the randomized data.

dataset & technique	*Co-clustering NCut*	*TD*	*Modularity*	*avg. L1*
Preserving the correlation distribution				
DBLP	**0.01**	**0.01**	**0.01**	1.00
Arxiv	**0.01**	0.31	**0.01**	0.06
Not preserving the corr. distrib. (α_{corr}=0)				
DBLP	**0.01**	**0.01**	**0.01**	**0.01**
Arxiv	**0.01**	**0.01**	**0.01**	**0.01**

Figure 6. Effect on p-value if avg. correlation distribution is not preserved

that the clustering result on Arxiv (w.r.t. TD) is primarily obtained due to the correlation of the neighbors. By not preserving the correlation, the clustering result (w.r.t. TD) is misleadingly assessed as interesting. Finally, in Fig 7 we randomize only one source of information. This can be easily done in our approach be setting $\Delta_{e/a} = 0$. Though, in any case, the correlation is preserved. As shown, when randomizing only the graph (i.e. the attributes remain unchanged), the results are sill significant. However, when randomizing the attributes (i.e. keeping the graph unchanged), some results become insignificant. This result again indicates that one source is often the dominant information used for clustering.

Figure 7. Effect on p-value if only one data source is randomized (DBLP data)

Randomization technique	*Shiga et. al. w. NCut*	*NCut*	*TD*	*Modul.*
default/both sources	**0.01**	**0.01**	**0.01**	**0.01**
only edges ($\Delta_a = 0$)	**0.01**	**0.01**	**0.01**	**0.01**
only attr. ($\Delta_e = 0$)	0.81	0.85	**0.01**	0.80

Recommendation & PageRank. Next, we assess results of recommendation algorithms and PageRank. We use the state-of-the-art recommendation algorithm SocialMF [6] for prediction in social networks with attribute (rating) data. As in [6] we predict the user ratings which were held out using 80-20 cross-validation, and we evaluate the rating performance via the well known root mean square error (*RMSE*). Finally, we apply weighted PageRank [24] to detect important vertices in the graphs. Weights on edges correspond to the similarity between two vertices (users) computed using Pearson correlation of the users' ratings. As structural measure we use the average PageRank value of the top-k highest ranked vertices. We analyze different values of k.

Fig. 8 shows the results on different data sets. Besides the p-values, we also include the actual values of the measures obtained by the methods, for example, the RMSE on the original data as well as its average and variance on the random data. As shown, SocialMF yields only significant results on the Epinions and Epinions$_{attr}$ data. If only the graph is given (Epinions$_{graph}$), the p-value is very large, meaning that the result is not significant. This demonstrates that the most valuable source for rating prediction is the attribute (rating) information. This is also confirmed by the absolute values of the RMSE: the best value is obtain for Epinions, followed by Epinions$_{attr}$, and then Epinions$_{graph}$. These results confirm the intuition behind matrix factorization based methods. In SocialMF the rating matrix is factorized to learn the latent factors for users and items, and these latent factors are used for rating prediction. The social network is only used to regularize the user latent factors and is helpful mostly for users with very few ratings. Therefore, we expect that if we loose the social network the accuracy loss will not be as large as if we loose the ratings.

dataset	*RMSE SocialMF p-val.*	*orig.*	*∅ rand*	*σ*	*avg. PageRank of top-10 vertices p-val.*	orig.	*∅ rand*	*σ*
Random	0.52	1.256	1.256	0.010	0.53	0.0134	0.0137	0.0014
Epinions	**0.01**	1.153	1.214	0.006	**0.01**	0.0222	0.0132	0.0022
Epinions$_{attr}$	**0.01**	1.168	1.256	0.010	1.00	0.0099	0.0137	0.0014
Epinions$_{graph}$	0.23	1.248	1.256	0.010	0.24	0.0146	0.0137	0.0014

Figure 8. Mining results of a recommender system and PageRank by using default randomization

In our experiments with the weighted PageRank method we get a different result. As shown in Fig. 8, only if both sources are maintained, significant results can be obtained. If the correlation between the sources is lost (e.g. since one source is random), we do not get significant results. Our experiment indicates that both network and attribute data are important. However, comparing the p-values for Epinions$_{attr}$ and Epinions$_{graph}$ we note that keeping the graph leads to a smaller p-value which implies the greater importance of the graph in the weighted PageRank method.

Fig. 9 presents the results for recommendation when using different randomization techniques. First, we compare our default randomization (both sources randomized, correlation preserved) with the one that does *not* preserves the correlation. In both cases the results are significant; however, by *not* preserving the correlation, the RMSE on the random data gets worse. Thus, the correlation is a meaningful source the recommender uses to increase its prediction quality. Second, we again randomize only one source of the data. Consider the average RMSE on the random data: if only the edges are randomized, the RMSE is close to the original data; when randomizing only the attributes, however, the RMSE drops more heavily. Again, we can infer that in this scenario the attributes are (in general) the more valuable source for recommendation. Please note, however, that the importance of the social network might vary between the users since the observed correlations result from different factors as, e.g., influence or homophily [8]. If influence is the main factor, the social network might be the more useful source for a *specific* user and item. Though, in any case using both sources simultaneously leads to the best prediction quality.

Randomization technique	*RMSE SocialMF p-val.*	*orig.*	*∅ rand*	*σ*
default	**0.01**	1.153	1.214	0.006
no corr. ($\alpha_{corr} = 0$)	**0.01**	1.153	1.256	0.010
only edges ($\Delta_a = 0$)	**0.01**	1.153	1.169	0.007
only attr. ($\Delta_e = 0$)	**0.01**	1.153	1.215	0.006

Figure 9. Effects by randomizing different information (Epinions data)

Finally, in Fig. 10 we perform a similar experiment for PageRank. We compute the average PageRank value over the top-k vertices for a range of values of k. We assess the significance if both sources, only the attributes, or only

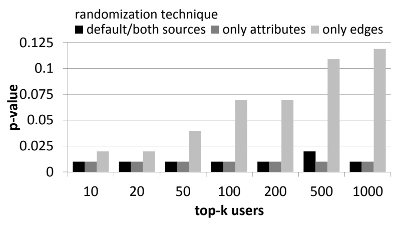

Figure 10. P-values for PageRank when considering varying number of vertices (Epinions data)

the edges are randomized. Again, in any case the correlation distribution of the input is preserved. Interestingly, the higher the number of considered vertices, the larger the p-value when randomizing only the edges (i.e. keeping the attributes unchanged). This might be explained as follows: in the original data (similar to most natural graphs) there exist a few highly ranked vertices that stand out and most of the other vertices have very low ranks. On the other hand, if we randomize the data, the ranks will be more evenly distributed. Now, if we only consider the very few top vertices (small k), they clearly stand out in the original data compared to the random data. However, by taking more vertices into account, this effect vanishes since more lower ranked vertices are considered in the top-k list.

Overall, all experiments demonstrate that by using both sources, graph and feature data, mining algorithms may gain a benefit. The correlation between the different sources is exploited, and for some combinations of data and algorithm the correlation might even be sufficient to describe one of the sources. In many cases, however, one source is dominant, and removing this source leads to a stronger decrease of the quality.

VI. Conclusion

We introduced a method to assess the significance of data mining results on graphs with binary feature vectors. We proposed to preserve the avg. correlation distribution, which inherently reflects the dependency between the graph and attribute data. Our randomization technique exploits an adaptive Metropolis sampling, interweaves attribute and graph randomization steps, and accurately preserves the proposed characteristics. We conducted thorough experiments on various real world datasets. Our findings indicate that combining the information from two different data sources can lead to more interesting mining results, i.e. the correlation between the sources is well exploited by the methods. Though, often one source has a larger impact on the final mining result.

As future work, we plan to extend our method to graphs with real-valued features and we will investigate the principle of Maximum Entropy for combined data sources.

Acknowledgment. This work has been supported by the UMIC Research Centre, RWTH Aachen University, Germany.

References

[1] M. Shiga, I. Takigawa, and H. Mamitsuka, "A spectral clustering approach to optimally combining numericalvectors with a modular network," in *KDD*, 2007, pp. 647–656.

[2] D. Hanisch, A. Zien, R. Zimmer, and T. Lengauer, "Co-clustering of biological networks and gene expression data," in *ISMB*, 2002, pp. 145–154.

[3] S. Günnemann, I. Färber, B. Boden, and T. Seidl, "Subspace clustering meets dense subgraph mining: A synthesis of two paradigms," in *ICDM*, 2010, pp. 845–850.

[4] S. Günnemann, B. Boden, and T. Seidl, "DB-CSC: A density-based approach for subspace clustering in graphs with feature vectors," in *ECML/PKDD (1)*, 2011, pp. 565–580.

[5] H. Ma, I. King, and M. R. Lyu, "Learning to recommend with social trust ensemble," in *SIGIR*, 2009, pp. 203–210.

[6] M. Jamali and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks," in *RecSys*, 2010, pp. 135–142.

[7] S.-H. Yang, B. Long, A. J. Smola *et al.*, "Like like alike: joint friendship and interest propagation in social networks," in *WWW*, 2011, pp. 537–546.

[8] A. Anagnostopoulos, R. Kumar, and M. Mahdian, "Influence and correlation in social networks," in *KDD*, 2008, pp. 7–15.

[9] M. McPherson, L. Smith-Lovin, and J. Cook, "Birds of a feather: Homophily in social networks," *Annual Review of Sociology*, vol. 27, pp. 415–444, 2001.

[10] A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas, "Assessing data mining results via swap randomization," *TKDD*, vol. 1, no. 3, 2007.

[11] M. Ojala, N. Vuokko, A. Kallio, N. Haiminen, and H. Mannila, "Randomization methods for assessing data analysis results on real-valued matrices," *Statistical Analysis and Data Mining*, vol. 2, no. 4, pp. 209–230, 2009.

[12] S. Hanhijärvi, G. C. Garriga, and K. Puolamäki, "Randomization techniques for graphs," in *SDM*, 2009, pp. 780–791.

[13] X. Ying and X. Wu, "Randomizing social networks: a spectrum preserving approach," in *SDM*, 2008, pp. 739–750.

[14] K.-N. Kontonasios, J. Vreeken, and T. D. Bie, "Maximum entropy modelling for assessing results on real-valued data," in *ICDM*, 2011, pp. 350–359.

[15] N. Vuokko and E. Terzi, "Reconstructing randomized social networks," in *SDM*, 2010, pp. 49–59.

[16] J. Besag and P. Clifford, "Generalized monte carlo significance tests," *Biometrika*, vol. 76, no. 4, pp. 633–642, 1989.

[17] M. Ojala, G. C. Garriga, A. Gionis, and H. Mannila, "Evaluating query result significance in databases via randomizations," in *SDM*, 2010, pp. 906–917.

[18] R. Fisher, "Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population," *Biometrika*, vol. 10, no. 4, pp. 507–521, 1915.

[19] T. Anderson, "On the distribution of the two-sample cramer-von mises criterion," *The Annals of Mathematical Statistics*, pp. 1148–1159, 1962.

[20] Y. Koren, R. M. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[21] C. Furlow and S. Beretvas, "Meta-analytic methods of pooling correlation matrices for structural equation modeling under different patterns of missing data." *Psychological Methods*, vol. 10, no. 2, p. 227, 2005.

[22] Y. Atchadé, G. Fort, E. Moulines, and P. Priouret, "Adaptive markov chain monte carlo: theory and methods," in *Bayesian Time Series Models*, 2011, pp. 32–51.

[23] N. Temme, "Asymptotic estimates of stirling numbers," *Stud. Appl. Math*, vol. 89, no. 3, pp. 233–243, 1993.

[24] C. C. Aggarwal and H. Wang, Eds., *Managing and Mining Graph Data*. Springer, 2010.