A Bayesian Method for Matching Two Similar Graphs without Seeds*

Pedram Pedarsani[1] and Daniel R. Figueiredo[2] and Matthias Grossglauser[1]

***Abstract*— Approximate graph matching (AGM) refers to the problem of mapping the vertices of two structurally similar graphs, which has applications in social networks, computer vision, chemistry, and biology. Given its computational cost, AGM has mostly been limited to either small graphs (e.g., tens or hundreds of nodes), or to large graphs in combination with side information beyond the graph structure (e.g., a seed set of pre-mapped node pairs). In this paper, we cast AGM in a Bayesian framework based on a clean definition of the probability of correctly mapping two nodes, which leads to a polynomial time algorithm that does not require side information. Node features such as degree and distances to other nodes are used as fingerprints. The algorithm proceeds in rounds, such that the most likely pairs are mapped first; these pairs subsequently generate additional features in the fingerprints of other nodes. We evaluate our method over real social networks and show that it achieves a very low matching error provided the two graphs are sufficiently similar. We also evaluate our method on random graph models to characterize its behavior under various levels of node clustering.**

I. Introduction

Mapping the vertices between two structurally similar graphs is known as approximate graph matching (AGM)[1], which has applications in computer vision, databases, and social networks, among others. For example, suppose two graphs represent similar but not identical social networks over the same vertex set. For example, one graph could represent an anonymized social network, while the other represents side information that an adversary might obtain. Matching the two graphs would then de-anonymize the first graph, thereby compromising user privacy [12].

AGM is a generalization of the classic graph isomorphism problem. In this problem, we seek a bijection π between the vertex sets of two graphs $G_{1,2}$ such that their edge sets $E_{1,2}$ are identical under π. The problem of identifying the isomorphic bijection is in NP [6] for general graphs, and there is no known polynomial time algorithm to compute it. Since AGM is at least as hard as the graph isomorphism problem, this might suggest that there is little hope in correctly matching large similar graphs.

*P. Pedarsani and M. Grossglauser were partially supported by the Swiss National Science Foundation (SNSF) under grant 200021-116510. D. Figueiredo was partially sponsored by grants FAPERJ, CNPq and CAPES (Brazil).

[1]IC, EPFL, Switzerland, {pedram.pedarsani, matthias.grossglauser}@epfl.ch

[2]Systems Engineering and Computer Science Dept. (PESC), Federal University of Rio de Janeiro (UFRJ), Brazil, daniel@land.ufrj.br

[1]The term *graph matching* in the literature also refers to the unrelated problem of selecting a set of edges of a single graph that have no common vertices.

Fortunately, it turns out that many realistic graphs have features that help AGM. For example, in pioneering work [12], Narayanan and Shmatikov devise a graph matching algorithm that succeeds in de-anonymizing a social network with millions of nodes, based on an initial *seed set* of correctly mapped nodes. Essentially, such methods exploit label-independent features of nodes that make them easy to identify in both graphs. Such features include the node degree, the clustering coefficient, or the volume (number of edges) in a neighborhood around the node.

The algorithm in [12] starts from an initial seed set of mapped node pairs, which may be obtained manually or through heuristics (e.g., identifying large cliques in both graphs). The algorithm then attempts to identify pairs of unmapped nodes that are neighbors of several seed nodes; among these pairs, it chooses the one that is best according to a heuristic based on node features, and adds it to the seed set. The algorithm then proceeds iteratively until the whole network is matched, or the process dies out. This requires (i) a seed set of sufficient size[2], and (ii) a sufficiently dense graph for the process to percolate. Recent results establish a phase transition in the seed set size [18] for the success of such an algorithm.

In this paper, we cast AGM in a probabilistic framework and propose an algorithm that relies on a rigorous model to infer the node map. The main advantages of this algorithm are the following:

- **Starting from scratch:** The algorithm does not require an initial seed set of mapped node pairs as input. The same probabilistic framework is used to build the map from start to finish.
- **Sparse graphs:** Our approach is particularly well suited to match sparse graphs. The algorithm is able to make progress even if the mapped nodes are far from each other.
- **Confidence metric:** Our method maintains an estimate of the likelihood of every mapped node pair. While this is used by the algorithm to build the map, it is also a very useful output for some applications, as it essentially provides a level of confidence for each mapped pair.

Our statistical framework and associated inference algorithm is based on the observation that in many real networks, some nodes possess features that make them stand out from

[2]Note that in their algorithm, if the seed nodes are all at distance ≥ 3 of each other, the process cannot start, as there is no candidate node that has more than one mapped neighbor.

the majority of other nodes; these nodes are much easier to map correctly than an average node. This suggests an incremental mapping process, where we first try to map the easiest nodes, and then use mapped nodes to map subsequent nodes. Each node has some fingerprint capturing label -independent metrics of interest. The matching relies on a cost function over fingerprints. Of course, errors are always possible, given that the two graphs are similar, but not identical. We deal with this by matching several node pairs at once through a maximum bipartite matching, rather than matching one node pair at a time. This is because the former takes into account the costs between all possible pairs in the set. Each phase of the algorithm matches a set of nodes, whose size doubles in every phase.

The second key contribution concerns the choice of fingerprint and the cost function used in the bipartite matching. We develop a probabilistic generative model for the two graphs that we are trying to match. This allows us to embed the matching problem into a clean Bayesian framework, which leads to a cost function over node pairs which can be interpreted as the logarithm of the posterior probability of a correct match given the fingerprints of the two nodes.

The paper is organized as follows. In Section II, we survey related work in graph matching. In Section III, we formally define the problem. Section IV describes the Bayesian framework underlying our matching algorithm, which is developed in Section V. Section VI reports experimental results over a real social network representing e-mail exchanges. Section VII contains a short conclusion.

II. Related Work

Approximate graph matching (AGM) has been investigated within and applied to several domains in the literature such as databases, pattern recognition, computer vision, and social networks. Within social networks, AGM has been applied to break anonymity by matching a labeled social network - where node identities are known - with an anonymized (unlabeled) social network. Recently proposed approaches to de-anonymize large social networks require side information in the form of a seed set (a set of known mapped node pairs) and/or manual intervention to bootstrap the algorithm, and are based on heuristics concerning how attributes of nodes can be used as evidence for finding the correct mapping [12], [11]. Recent work also sheds light on the performance of this approach, and establishes a sharp phase transition in the size of the seed set between a regime where the algorithm almost certainly fails, and a regime when it succeeds in matching almost everything [18].

Other approaches have focused on de-anonymizing only specific individuals or small fractions of the social network using more rigorous probabilistic frameworks, such as random graph models [1], [8], [17]. Our work provides a Bayesian framework for AGM that can be applied for de-anonymization of social networks without requiring any side information. In fact, we evaluate our method using real social networks (see Section VI).

Within the database literature, AGM has a different flavor: given a query graph G, the goal is to search a large graph database for graphs that are similar to G (or that have a subgraph that is similar to G). Since mapping nodes of the query graph G to nodes of graphs in the database is too expensive, most approaches rely on indexing features of graphs (and their subgraphs) in the database to find graphs similar to G [15], [20]. Thus, the problem investigated in this domain is different from ours.

Graph matching has a long history within the pattern recognition literature (see the survey [3]). Several works focus on exact graph matching, which is equivalent to the graph isomorphism problem (or exact subgraph matching). The goal is to find a edge-preserving node mapping [4], [16]. A variation of this problem finds an almost correct (approximate) node mapping between the graphs using statistical methods, such as random walks [7], expectation-maximization (EM) [9], and the Bayesian edit-distance [10].

Another recent body of work addresses AGM in the same setting as considered in this paper. However, these approaches rely on side information in the form of label-dependent features of nodes and/or edges when defining a similarity measure between two nodes. A global cost function measuring the quality of the mapping is then maximized using optimization and linear programming methods [5], [14], [2], [19]. In our setting, the similarity measure has to be label-independent.

III. Problem Definition

We consider the problem of matching the vertex sets of two graphs. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ denote two graphs with the same number of nodes, $|V_1| = |V_2| = n$, but different edge sets. Let $\pi : V_1 \rightarrow V_2$ be a mapping that establishes a one-to-one correspondence between nodes of G_1 and G_2. Assume there exists a correct mapping π_0 between $u_1 \in V_1$ and $u_2 \in V_2$. The problem we consider is determining π_0 using as input only $G_{1,2}$ and no other labeling information.

Finding the correct mapping strongly depends on the relationship between the structure of $G_{1,2}$. For example, the correct mapping cannot be attained if one of the edge sets has no (or very few) edges, since no structural information may remain to be explored. Moreover, the problem is hard even if the two graphs are structurally the same, since this reduces to finding an isomorphism between the two graphs, a well-known NP problem (not NP-complete, however) for which no polynomial-time algorithm is known to exist.

For ease of notation, and without loss of generality, we assume from now on that $V_1 = V_2 = V$, and that the true mapping is given by the identity permutation. Of course, this notational assumption is not visible to the matching algorithm.

IV. Method for Matching Two Similar Graphs

Conceptually, matching two graphs can be cast as minimizing a cost function over all $n!$ possible matchings π of the vertex sets of the two graphs $G_{1,2}$ [13]. As the

computational cost is prohibitive for graphs of non-trivial size, we seek a method to reduce the search space for π. We achieve this by building π in $\log_2 n$ phases. In each phase $\tau = 0, \ldots, \log_2 n - 1$, we double the number of mapped nodes, which we refer to as *anchors*, from the previous phase, until we have mapped every node. We let S_τ denote the set of anchors produced at the end of phase τ. As will become clear, this map is not necessarily monotonically increasing (i.e., $S_\tau \not\subset S_{\tau+1}$ in general): a node pair can be an anchor in one phase, but be unmapped at a later phase.

We reduce the complexity of the problem by summarizing the structure of the two graphs $G_{1,2}$ as a set of (label-independent) attributes for each node. Then we rely on these node attributes to match 2^τ nodes in each phase τ, used as anchors for the next phase, where this matching is only a function of the attributes. We call the set of attributes of a node its *fingerprint*, and rely on a cost function over pairs of fingerprints to compute the matching. The matching algorithm is described in detail in the next section; in this section, we define the node fingerprints, and the probabilistic model that yields the cost function over pairs of fingerprints, and its Bayesian interpretation.

We first require a notion of similarity between two graphs $G_{1,2}$. The ultimate goal is to obtain a measure of similarity between two node fingerprints. We assume that the observable graphs $G_{1,2}$ are obtained through a sampling process on a (fixed) generator graph $G = (V, E)$. We elaborate on the choice of this process in Section IV-C.

In the following, we first formalize the computation of the probability of two nodes being a correct mapping, and describe one step of this iterative procedure assuming some nodes have already been mapped. We present the complete iterative algorithm that incrementally increases the set of mapped nodes in Section V. As later discussed, the first phase of the algorithm requires no knowledge of any anchor nodes.

A. Pairwise Posteriors

Consider the two observed graphs G_1 and G_2 and two nodes $u_1 \in V_1$ and $u_2 \in V_2$ with their fingerprints

$$\begin{aligned} F_{u_1} &= [X_{11}, \ldots, X_{1,\ell}, X_{1,\ell+1}, \ldots, X_{1,\ell+m}] \\ F_{u_2} &= [X_{21}, \ldots, X_{2,\ell}, X_{2,\ell+1} \ldots, X_{2,\ell+m}], \end{aligned} \quad (1)$$

where each component is a structural attribute that provides evidence for the similarity of u_1 and u_2. More specifically, the components $1, \ldots, \ell$ are *absolute* attributes, in that they depend only on the graphs G_1 and G_2, respectively. They may include local graph statistics around node $u_{1,2}$, such as the node degree, the clustering coefficient, or the density of some neighborhood.

A key idea in our method is to rely on anchor nodes from the previous phase to obtain additional attributes for the current phase. In this way, the matching of nodes in later phases can build on nodes matched earlier. The components $\ell+1, \ldots, m$, with $m = |S_{\tau-1}| = 2^{\tau-1}$ are *relative* attributes, which depend both on $G_{1,2}$ and the anchor pairs $S_{\tau-1}$ from the previous phase. They may include different distance metrics between $u_{1,2}$ and each anchor pair, such as hop distance or resistor distance. If two nodes $u_{1,2}$ have similar distances from a set of anchors, this is strong evidence that they match, *provided the anchors themselves are matched correctly.*

We now turn to the fingerprint cost function for the node pair $u_{1,2}$ which is simply given by the probability that the two nodes correspond to the same node in G given the two fingerprints, i.e., the posterior probability, which we write as

$$r(u_1, u_2) = P\left[U_1 = U_2 | F_{u_1}, F_{u_2}\right]. \quad (2)$$

Recall that the matching algorithm assumes node labels provide no information towards matching. We model this by considering that the algorithm is looking at two randomly chosen nodes $U_1 \in V_1$, $U_2 \in V_2$, which are independent uniform random variables over $[1, n]$. Thus, in the absence of any node attributes, the probability that U_1 and U_2 correspond to the same node in G is given by $1/n$, leading to a uniform prior[3] for correct matching, $P[U_1 = U_2] = 1/n$.

Using Bayes' rule and the observation above, we can rewrite (2) to obtain:

$$r(u_1, u_2) = \frac{P\left[F_{u_1}, F_{u_2} | U_1 = U_2\right] 1/n}{P\left[F_{u_1}, F_{u_2} | U_1 = U_2\right] 1/n \cdots + \cdots + P\left[F_{u_1}, F_{u_2} | U_1 \neq U_2\right] (1 - 1/n)} \quad (3)$$

Of course, different node attributes may be correlated in complicated ways, depending on the underlying generator G. For example, a high-degree node could have, on average, smaller distances to other nodes in the graph than a low-degree node. However, we assume that pairs of corresponding attributes of two nodes U_1 and U_2 are conditionally independent, given either $U_1 = U_2$ or $U_1 \neq U_2$. For the latter case $U_1 \neq U_2$, this is because the attributes stem from two random, but different nodes in G; for the case $U_1 = U_2$, this embodies the assumption that the sampling process that generates $G_{1,2}$ acts on each attribute pair independently. Although this is a strong assumption, it does not prevent our approach from accurately matching nodes, as illustrated by our results (see Section VI). Under this assumption, we can factor the evidence in (3) and obtain

$$r(u_1, u_2) = \frac{1/n \prod_{i=1}^{\ell+m} P\left[X_i | U_1 = U_2\right]}{1/n \prod_{i=1}^{\ell+m} P\left[X_i | U_1 = U_2\right] \cdots + \cdots + (1 - 1/n) \prod_{i=1}^{\ell+m} P\left[X_i | U_1 \neq U_2\right]}. \quad (4)$$

where $X_i = (X_{1i}, X_{2i})$ is a pair of corresponding attributes. We do not assume that a given pair of corresponding evidence is independent when the two nodes correspond to one another, namely $P[(X_{1i}, X_{2i}) | U_1 = U_2] \neq P[X_{1i} | U_1 = U_2] P[X_{2i} | U_1 = U_2]$. In fact, the evidence pair is likely to be correlated given that the two nodes correspond to one another.

In order to apply (4) we need to determine the terms $P\left[X_i | U_1 = U_2\right]$ and $P\left[X_i | U_1 \neq U_2\right]$ for all $i = 1,$

[3]Of course, other priors are possible, e.g., if additional side information about nodes is available, such as from additional node attributes.

$\ldots, \ell+m$. This probability depends on the sampling process, as the sampling process *transforms* attributes of G. Let $p_i(y)$ denote the probability distribution associated with attribute i in the fixed graph G, thus, $p_i(y) = P[Y_i = y]$, for some node $u \in V$. Similarly, let $q_i(x, y)$ denote the probability function of attribute i after the sampling process has been applied to G given that the same attribute has value y in G, thus, $q_i(x, y) = P[X_{1i} = x|Y_i = y]$, for some $u \in V$. Using these two models together with the fact that G_1 and G_2 are independent samples from the sampling process and by applying the law of total probability, we obtain

$$P[X_i|U_1 = U_2] = \sum_y q_i(x_{1i}, y)q_i(x_{2i}, y)p_i(y). \quad (5)$$

Likewise, for the case u_1 and u_2 do not correspond to the same node in G, we have, $P[X_i|U_1 \neq U_2] =$

$$\left(\sum_y q_i(x_{1i}, y)p_i(y)\right)\left(\sum_y q_i(x_{2i}, y)p_i(y)\right). \quad (6)$$

In this case the corresponding evidence is independent since the nodes do not correspond to one another and where chosen uniformly at random from $G_{1,2}$, thus, $P[(X_{1i}, X_{2i})|U_1 \neq U_2] = P[X_{1i}]P[X_{2i}]$.

Finally, using the Bayesian framework above, together with the models for how the sampling process transforms given attribute ($q_i(x, y)$), we can precisely compute the probability that a given pair of nodes is correctly mapped given their fingerprints (the posterior probability).

B. Handling Anchor Mismatch

The approach above allows for the set of anchor nodes mapped in the previous phase to be used to compute the posterior probability for the unmapped nodes. However, anchors are subject to matching errors. If we do not take this into account, we may overestimate the posterior probabilities. Therefore, we refine our probabilistic model to account for the possibility of wrong matches among anchor pairs.

Recall that $X_{\ell+1}, \ldots, X_{\ell+m}$ depend on the anchor pairs. For example, $X_{\ell+i}$ is the distance from u_1 and u_2 to the i-th anchor pair w_{1i} and w_{2i}, respectively.

Let M_i be an indicator random variable taking value 1 if anchor pair $\ell + i$ has been correctly mapped and 0 otherwise. We now revisit (5), and assume that if $M_i = 0$, the distribution of the corresponding attribute pair is as if $U_1 \neq U_2$. In other words, we assume that the distance pair from a node ($u_1 = u_2$ or $u_1 \neq u_2$) to an anchor pair that *do not correspond to one another* is independent, in analogy to the assumption on the distance pair from two different nodes $u_1 \neq u_2$ to a correctly matched anchor pair. Thus, we have

$$P[X_{\ell+i}|U_1 = U_2, M_i = 0] = \left(\sum_y q_{\ell+i}(x_{1(\ell+i)}, y)p_{\ell+i}(y)\right)\left(\sum_y q_{\ell+i}(x_{2(\ell+i)}, y)p_{\ell+i}(y)\right) \quad (7)$$

This is the same as the conditional probability given that $U_1 \neq U_2$ (see (6)). We also assume that a correctly or incorrectly mapped anchor does not change the conditional probability when $U_1 \neq U_2$. Thus,

$$P[X_{\ell+i}|U_1 \neq U_2, M_i] = P[X_{\ell+i}|U_1 \neq U_2] \quad (8)$$

as in (6).

In order to compute the overall pairwise posterior, we redefine the probability function of (2) to also depend on anchors being correctly mapped. Thus, $r(u_1, u_2) =$

$$P[U_1 = U_2|X_1, \ldots, X_{\ell+m}, M_1, \ldots, M_m] = P\left[U_1 = U_2|X_1, \ldots, X_\ell, (X_{\ell+1}, M_1), \ldots, (X_{\ell+m}, M_m)\right]. \quad (9)$$

As before, we apply Bayes' rule together with conditional independence between attributes pairs to obtain

$$r(u_1, u_2) = \frac{A}{A + B} \quad (10)$$

where,

$$A = 1/n \prod_{i=1}^{\ell} P[X_i|U_1 = U_2] \prod_{\substack{i=\ell+1 \\ \text{s.t. } M_i=1}}^{\ell+m} P[X_i|U_1 = U_2], \quad (11)$$

and

$$B = (1 - 1/n) \prod_{i=1}^{\ell} P[X_i|U_1 \neq U_2] \prod_{\substack{i=\ell+1 \\ \text{s.t. } M_i=1}}^{\ell+m} P[X_i|U_1 \neq U_2]. \quad (12)$$

All the factors where $M_i = 0$ appear both in the numerator and denominator (in A and B) and therefore cancel out. Moreover, (10) is for a given vector of M_i values.

Finally, note that $M_i, i = 1, \ldots, m$ are not observable, since we do not know if an anchor was correctly or incorrectly mapped. We marginalize out the $\{M_i\}$ to obtain

$$P[U_1 = U_2|X_1, \ldots, X_{\ell+m}] = \sum_{b=(0,1)^m} \prod_{i=1}^{m} (P[M_i = b_i]) \cdot P[U_1 = U_2|X_1, \ldots, X_{\ell+m}, M_1, \ldots, M_m]. \quad (13)$$

As the sum is over an exponentially large set (2^m possible choices of the bit vectors of size m), we use a simple Monte Carlo estimator of (13), by generating samples of vectors $[M_1, ..., M_m]$. In the experiments reported later we used 50 samples, which yielded satisfactory results.

Finally, we require the prior on M_i in (13). Fortunately, this can be obtained as a byproduct of the previous phase: for an anchor pair (w_{1i}, w_{2i}) matched in the previous phase, we equate its posterior $r(w_{i1}, w_{i2})$ with the prior $P[M_i = 1]$ in the new phase, as this is the probability that anchors w_{1i} and w_{2i} were correctly mapped. We discuss this procedure in more detail in Section V. Figure 1 depicts the Bayesian network of our problem formulation, indicating absolute and relative attributes. Relative attributes depend on anchor nodes and that the number of relative attributes depends on the phase of the algorithm, as described in Section V.

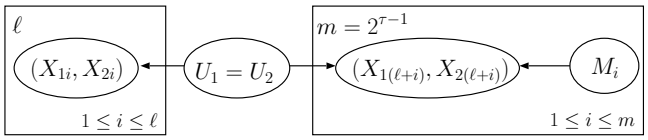


Fig. 1. Graphical model for the dependence of anchor-independent (left) and relative (right) feature pairs. M_i is an indicator for pair i to be correctly matched. $U_1 = U_2$ if the feature pair stems from the same underlying generator node.

C. Models for Sampled Degree and Distance

In order to apply the above methodology we must specify node attributes used in their fingerprints as well as probabilistic models that characterize such attributes. From now on, we will use the node degree as the only absolute attribute, thus, $\ell = 1$, and hop distances from the node to anchors as the relative attributes. Thus, $X_1 = (X_{11}, X_{21})$ is the degree pair of nodes $u_{1,2}$, and $X_{1+i} = (X_{1(1+i)}, X_{2(1+i)})$, $i > 0$ is the distance pair to anchor pair i[4]. Having established the attributes to be used in the fingerprint of nodes, we now need probabilistic models that represent both degree and distances in G (the generator graph) as well as in $G_{1,2}$ (the observed graphs). Note that these models are used directly in equations (5) and (6).

We consider the *edge sampling* process used in [13] to generate G_1 and G_2 from a fixed graph G. In edge sampling, each edge from G is sampled independently and with the same probability s. Thus, G_1 and G_2 are independent outcomes of edge sampling from G with probability s (in the general case, s_1 and s_2 can be used as sampling probabilities for G_1 and G_2, respectively). Note that the sole parameter s controls the amount of structure from G appearing in the sampled graph and that s^2 is the edge overlap probability (i.e., an edge of G appearing in both G_1 and G_2), which can be interpreted as a measure of similarity between the two graphs.

Under the edge sampling process, we can now specify a model for degree and distance on the sampled graph given their values in the original graph G. Recall that $q_i(x, y)$ denotes the probability that attribute i has value x in the observed graph, given that it has value y in the original graph G. As mentioned above, $i = 1$ denotes the node degree. As the edge sampling process samples each edge independently with probability s, $q_1(x, y)$ follows a binomial distribution with parameters y and s, i.e.,

$$q_1(x, y) = \binom{y}{x} s^x (1 - s)^{y-x}$$

Finally, note that $p_1(y)$ denotes the degree distribution of G, i.e., the probability that a node in G has degree y. We comment on the choice of the underlying degree distribution in Section VI.

An exact model for distances under edge sampling is much more complicated due to dependencies of paths in the graph. We therefore propose the following idea for an approximate model for how distances increase: for each hop in the original shortest path in G, an extra hop is taken in the sampled graph with probability $1 - s$. Thus, the distance in the sampled graph is given by a constant (the original distance y) plus a binomial random variable with parameters y and $1 - s$, i.e, $Y = y + \mathsf{Bi}(y, 1 - s)$, where Y is the distance in the sampled graph. Let $q_{1+i}(x, y)$ denote the probability that the distance to anchor node $i, 1 \le i \le m$ is x given that this distance in G is y. We then have,

$$q_{1+i}(x, y) = \binom{y}{x - y} (1 - s)^{x-y} s^x \quad , \quad y \le x \le 2y$$

Although the above model is a heuristic approximation for how distances enlarge under edge sampling, results shown in Section VI support its adequacy. Moreover, experiments with other approximate distance models as well as empirical distance distributions yielded similar results. Finally, $p_{1+i}(y)$ for $1 \le i \le m$ corresponds to the distance distribution of G, i.e., the probability that two given nodes are at distance y in G. We comment on the choice of the underlying distance distribution in Section VI.

Note that any attribute that is used in the fingerprint of the nodes will require a probabilistic model that describes how the attribute is transformed from G into the observed edge sampled graph $G_{1,2}$, namely, $q(x, y)$ as defined above degree and distance. Moreover, we will also need a probabilistic model for the prior distribution of the attribute in G, namely $p(y)$ as defined above. Using just degree and distances requires just two models, both of which are quite simple (in fact, it is exact for degree). This is one reason for using just these two attributes. Moreover, by using distances to various nodes (i.e., anchors) we can generate multiple attributes of the same type, therefore using the same probabilistic model, as we assume conditional independence. Finally, we observe experimentally that these two attributes perform relatively well when considering both real world networks and random graph models with clustering (details in Section VI).

V. Matching Algorithm

The matching algorithm receives as input two graphs G_1 and G_2, with the same number of nodes and models for distance and degree as described in Section IV-C. The output of the algorithm is a matching π between all nodes such that $\pi(u_1) = u_2$ where $u_1 \in V_1$ and $u_2 \in V_2$. The algorithm is iterative and at each phase it considers a set of candidate nodes to be matched, denoted by

$$\begin{aligned} V_1^\tau &= \{u_{11}, u_{12}, \ldots, u_{1n_\tau}\}, \quad u_{1i} \in V_1 \\ V_2^\tau &= \{u_{21}, u_{22}, \ldots, u_{2n_\tau}\}, \quad u_{2i} \in V_2 \end{aligned}$$

where n_τ is the size of the candidate set in the τ-th phase, $\tau = 0, 1, \ldots, \log_2 n - 1$, and more specifically, $n_\tau = 2^{\tau+1}$. In particular, in phase τ, the set of candidate nodes are simply the n_τ nodes with the largest degrees in their respective graph, thus $d(u_{ki}) \ge d(u_{kj}), i \le j$, where $d(u)$ is the degree of node u.

[4]More precisely, with (w_{1i}, w_{2i}) the i-th anchor pair, $X_{1(1+i)}$ is the hop distance in G_1 to w_{1i} (and analogously for $X_{2(1+i)}$).

We elaborate on one phase of the algorithm. At phase τ, we consider a set of $m = 2^{\tau-1}$ anchors (previously mapped node pairs) $S_{\tau-1} = \{(w_{11}, w_{21}), \ldots, (w_{1m}, w_{2m})\}$. Note that we set $m = 0$ in the first phase ($\tau = 0$), and the algorithm requires no prior knowledge of anchor nodes. For each of the n_τ^2 node pairs (u_{1i}, u_{2j}) in the candidate set, we use the Bayesian framework described in Section IV to compute their matching posterior $r(u_{1i}, u_{2j})$ by using their fingerprints. In particular, as discussed in Section IV-C, we use as evidence the degree pair $(d(u_{1i}), d(u_{2j}))$ and m distance pairs $(x(u_{1i}, w_{1k}), x(u_{2j}, w_{2k}))$, $k = 1, \ldots, m$, where $x(u, v)$ is the distance between nodes u and v in their respective graph.

Assuming that pair-wise matchings of nodes are independent given a matching (i.e., every node has a single pair), the posterior terms $r(u_{1i}, u_{2j})$ can be factorized, and the probability of a particular mapping π is given by the product of all pair-wise posteriors under that mapping, i.e.,

$$P_\pi(V_1^\tau, V_2^\tau) = \prod_i^{n_\tau} r(u_{1i}, \pi(u_{1i})), \tag{14}$$

We seek the most likely mapping $\pi_* = \arg\max_\pi P_\pi(V_1^\tau, V_2^\tau)$, i.e., the mapping that maximizes the product in (14) in the τ-th phase. Taking the logarithm of each posterior term $r(u_{1i}, u_{2j})$, the problem reduces to maximizing the sum of log-posteriors for all node pairs:

$$\pi_* = \arg\max_\pi \sum_i^{n_\tau} \log r(u_{1i}, \pi(u_{1i}))$$

This problem can then be solved by using the Hungarian algorithm for maximum weighted bipartite matching [6].

Finally, the matching results from phase τ are used to construct the set of anchor node pairs S_τ (the nodes that were mapped with highest posteriors) for the next phase, where $|S_\tau| = 2^\tau$. In other words, half of the mapped node pairs are considered as anchors for the next phase $\tau + 1$. This increases the size of the fingerprint of each node in the candidate set with more distance attributes. As $n_\tau = 2^{\tau+1}$, the size of the candidate set of nodes is also doubled in the next phase.

The intuition for choosing half of mapped pairs as anchors is that matching errors tend to increase in the ordered (high-to-low posterior) sequence of mapped pairs. In particular, it is not even guaranteed that a candidate node $u_{1i} \in V_1^\tau$ has a correct match in V_2^τ, and this gives rise to more uncertainty in matching pairs with low posteriors. Thus, we use only the top half of matching pairs as anchors of the next phase, aiming to increase the confidence of the matching. A detailed description of the matching algorithm follows:

1) Input: two graphs G_1 and G_2, with node sets V_1 and V_2, both of size n, and models for degree and distance.
2) Sort the nodes in each graph by degree in descending order. Let V_1' and V_2' be the list of nodes in descending order by degree.
3) Set $\tau = 0$. Set the anchor set as $S_{\tau-1} = \emptyset$.
4) Set the size of the candidate set to be mapped as $n_\tau = 2^{\tau+1}$. If $n_\tau > n$, set $n_\tau = n$.
5) Denote the candidate sets as $V_1^\tau = V_1'(1 : n_\tau)$ and $V_2^\tau = V_2'(1 : n_\tau)$, as the list of the first n_τ nodes in V_1' and V_2', respectively.
6) For each node $u_{1i} \in V_1^\tau$ and $u_{2j} \in V_2^\tau$, compute a fingerprint vector as

$$\begin{aligned} F_{u_{1i}} &= [d_{u_{1i}}, x(u_{1i}, w_{11}), \ldots, x(u_{1i}, w_{1m})] \\ F_{u_{2j}} &= \left[d_{u_{2j}}, x(u_{2j}, w_{21}), \ldots, x(u_{2i}, w_{2m})\right], \end{aligned}$$

   where $i, j = 1, \ldots, n_\tau$ and $(w_{1k}, w_{2k}) \in S_{\tau-1}$, $k = 1, \ldots, m$.
7) For each pair of nodes $(u_{1i}, u_{2j}) \in V_1^\tau \times V_2^\tau$, compute the matching posterior $r(u_{1i}, u_{2j})$, i.e., the posterior $P[u_{1i} = u_{2j} | F_{u_{1i}}, F_{u_{2j}}]$, using (13). Construct the complete weighted bipartite graph $G(V_1^\tau, V_2^\tau, E)$, where E includes n_τ^2 edges between all node pairs $(u_{1i}, u_{2j}) \in V_1^\tau \times V_2^\tau$, with the edge weights equal to $r(u_{1i}, u_{2j})$.
8) Compute a *normalized* posterior $r'(u_{1i}, u_{2j})$ as follows:

$$r'(u_{1i}, u_{2j}) = \frac{r(u_{1i}, u_{2j})}{\sqrt{\sum_{v \in V_2} r(u_{1i}, v) \ \sum_{v \in V_1} r(v, u_{2j})}}. \tag{15}$$

9) Solve the maximum $r'(.,.)$-weighted bipartite matching, e.g., using the Hungarian algorithm [6]. The output is the matching π of size n_τ.
10) Sort the matched pairs $(u, \pi(u))$ in descending order of their matching posterior. Let S_τ be the set of $n_\tau/2 = 2^\tau$ node pairs with highest posteriors. Set $m = |S_\tau|$.
11) If $n_\tau = n$, stop. Else, increment τ. Go to step 4.

The algorithm starts with the two highest degree nodes in each graph ($n_\tau = 2$), computes node fingerprints based only on their degrees, and matches them using the bipartite matching. Next, half of the matched nodes (i.e., 1 node pair), are considered as anchors, and the size of the candidate set is doubled to $n_\tau = 4$. Thus, at the second phase, four nodes are matched by using their degrees and distance to one anchor from the previous round. Again, half of the matched nodes (i.e., 2 node pairs), are considered as anchors for the next round, and the size of the candidate set is doubled to $n_\tau = 8$. This process continues, and at each phase, n_τ nodes are matched using $n_\tau/4$ anchors, until all nodes are mapped. Notice that the matched nodes at each phase, though being considered as anchors for the next phase, are themselves still in the candidate sets V_1^τ and V_2^τ. This allows for a node to change its mapping during the execution of the algorithm. This is useful because a node wrongly mapped in an early phase might be correctly mapped in a subsequent phase when more evidence (in terms of distance to anchors) is available.

The complexity of the algorithm above is $O(n^3 \log n)$, as the algorithm takes $\log n$ phases and the Hungarian algorithm has complexity $O(kn)$, where k is the number of edges in the bipartite graph, which in our case is n^2 in the last phase. However, the algorithm can be modified to have complexity $O(n^2 \log^2 n)$ by considering a bipartite graph that is not

complete (all edges), but rather sparse. We do not discuss the details of the process in this paper. Finally, note that the algorithm requires computing the distances between nodes and anchors in both $G_{1,2}$. However, the distances between all pairs of nodes in both $G_{1,2}$ can be computed before starting the algorithm. This has a complexity of $O(n(|E_1| + |E_2|))$ since a BFS for each node in $G_{1,2}$ will suffice.

The intuition behind the normalization (step 8 of the algorithm) is the following. Recall that we assume independence between the posteriors of node pairs, which ensures equivalence with the weighted bipartite matching problem. However, this assumption can lead to an overestimation of the joint node pair posteriors. The normalization procedure compensates for this effect by reducing the value for the posteriors. Furthermore, experimental results show that this normalization yields significantly better results in matching all node pairs.

Finally, we reiterate that in the problem formulation (see Section III) as well in the above algorithm, we have assumed not only that the number of nodes in the two graphs $G_{1,2}$ is the same, but that every node in a graph has a corresponding node in the other graph. Of course, in many practical scenarios, we would expect to have only partial overlap in the vertex sets. One way to deal with partially overlapping graphs would be to refrain from mapping node pairs that have posteriors below a certain threshold. We have not evaluated such strategies for partial matching, but our experiments in the next section suggest that a low posterior is indeed a good indication of a poor match.

VI. Experiments

We assess the performance of our algorithm using real data and two different scenarios. First, we consider a fixed graph G and use the edge sampling model with probabilities s_1 and s_2 to instantiate G_1 and G_2, which are then matched. Next, we consider two *similar* graphs G_1 and G_2 from real data, which are then matched. Note that sampling was not used in the latter and parameters will have to be estimated using G_1 and G_2 directly. We also assess the performance using realizations from random graph models (Section VI-D). To measure the performance we define the matching error as the fraction of node pairs that have been mapped.

A. Matching Two Sampled Graphs

We consider a dataset of e-mail messages collected at the EPFL main mail server. The dataset includes logs of e-mail exchanges among users on a weekly basis and we consider only exchanges *among EPFL users*. We construct the e-mail network (graph) by associating each user to a node and placing an edge between two nodes if there is at least one reciprocal e-mail exchange between the two users. We consider a period of five consecutive weeks and only nodes that appeared (i.e., send/received a message) in every week, yielding a graph with 2024 nodes and 25603 edges.

Using the graph generated as the underlying graph G, we apply edge sampling to obtain G_1 and G_2. For simplicity, we assume $s_1 = s_2 = s$ and present our results for different values of s^2, which corresponds to the probability that an edge in G appears in both G_1 and G_2 (edge overlap) and is a measure of similarity between the two graphs. For each s, we repeat the sampling process to obtain different realizations of G_1 and G_2. To compute the posteriors, we use the empirical degree and distance distributions of G for $p_i(y)$. Moreover, as we know s, the conditional distribution $q_i(x, y)$ is computed using the degree and distance models presented in Section IV-C.

Figure 2 depicts the matching error as the algorithm progresses. Each plot corresponds to a fixed value of s^2 and each curve corresponds to one realization of G_1 and G_2. The results indicate that the performance of the algorithm at the end (after all nodes are matched) is consistently poor and good for small and large values of s^2 ($s^2 = 0.49$, $s^2 = 0.81$), respectively. The curves also indicate an interesting aspect of our approach: although the error might be high during the first iterations (due to incorrect anchors), the algorithm remaps such pairs as it progresses and achieves a smaller error at the end. To some extent, our approach does not depend on correctly identifying initial anchors and is therefore robust to initial errors.

Figure 3(a) (*Sampled* curve) shows the average final matching error (over 20 realizations) and 95% confidence interval versus the edge overlap s^2. As shown, the error is relatively high for small overlap $s^2 = 0.49$ since the structure of G is significantly destroyed. However, the error drops sharply when we increase s^2, reaching 6% for $s^2 = 0.81$. We can also notice the larger confidence intervals for mid-range s^2, indicating that in such regimes the matching error strongly depends on the realization.

Recall that at the end of the algorithm, each mapped node pair has a posterior probability of being correctly mapped, which can be used to assess the quality of the mapping. Figure 3(b) depicts the empirical CCDF (Complementary Cumulative Distribution Function) of the posteriors $r(u_{1i}, u_{2j})$ for all n^2 pairs of nodes in $V_1 \times V_2$ for (bottom curve), and all n pairs of nodes mapped at the end of the algorithm (top curve). Note that other curves correspond to result for the $G(n, p, q)$ model and will soon be described.

The clear distinction between the two curves for EPFL indicates the effectiveness of our method: Although most node pairs have very small posteriors, a small subset exhibit high posteriors among which are the pairs that correspond to each other. For instance, 60% of the mapped node pairs have a posterior greater than 0.8, whereas the probability of such posterior over all nodes is slightly above 0.001. In other words, the posterior constructed through our Bayesian method is an adequate metric for the similarity between two nodes. This is the key reason why the maximum bipartite matching resolves the matching problem with minimal error. Moreover, the posterior probability can be used to assess the quality of the pairwise mapping, since it indicates the probability that the pair is correctly mapped, given all the evidence.

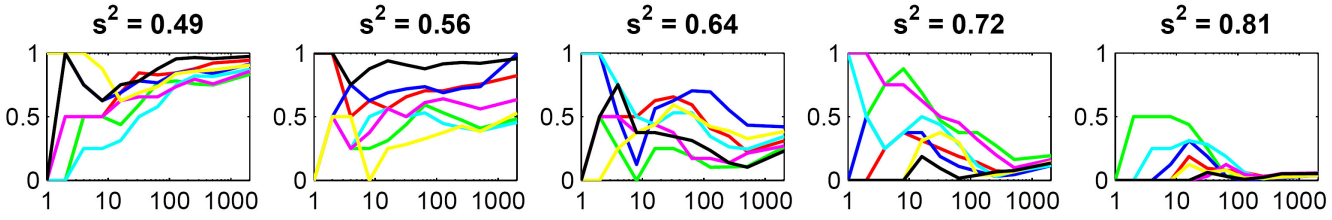
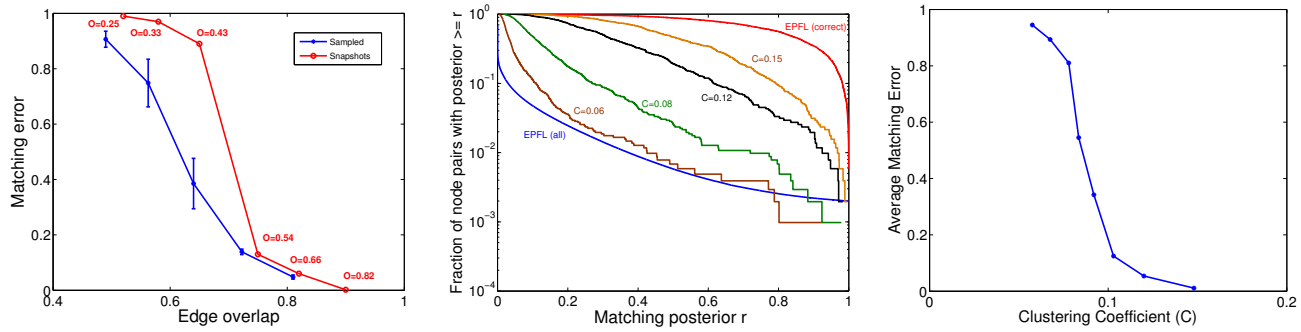


Fig. 2. Matching error as a function of the number of mapped nodes (as the algorithm progresses in rounds) for different edge overlap probabilities (s^2) using EPFL e-mail network as the fixed underlying graph. Each curve in a plot corresponds to a realization of the edge sampling process, namely different instances for $G_{1,2}$.



(a) (b) (c)

Fig. 3. (a) Matching error as a function of edge overlap: *Sampled* curve corresponds to two realizations of the edge sampling model from a real EPFL e-mail network (error averaged over 20 realization pairs), *Snapshots* curve corresponds to two real EPFL E-mail Networks generated with overlapping time windows (single realization); (b) Empirical CCDF for posteriors of matched node pairs and all node pairs for EPFL e-mail network ($s^2 = 0.81$), and $G(n, p, q)$ model for different clustering coefficients (C values near curves) but constant average degree (50); (c) Average matching error in $G(n, p, q)$ model for different clustering coefficients (C) but constant average degree (50).

B. Matching Two Graph Snapshots

We now match two real snapshots G_1 and G_2 from EPFL e-mail network without considering any fixed underlying graph G. We construct G_1 by considering the set of common nodes over a window of 10 weeks and all edges among them. The graph has 1825 nodes and 24196 edges. The graph G_2 is constructed in the same manner by also considering a period of 10 weeks but with a time shift of t weeks relative to G_1. For example, $t = 3$ means that G_1 and G_2 overlap 7 weeks, thus giving $O = 7/(10 + 3) = 54\%$ time overlap. Note that time overlap translates to edge overlap. In particular, we compute the parameters s_1 and s_2 based on the number of edges in G_1, G_2, and their intersection. We also compute the empirical degree and distance distributions for the (unknown) underlying graph by superposing the distributions in G_1 and G_2 (see Section VI-C). Finally, we apply our algorithm to match the common nodes of G_1 and G_2.

Figure 3(a) (*Snapshot* curve) depicts the matching error as a function of edge overlap between G_1 and G_2. Each point in the curve corresponds to a different time overlap O, as indicated. Note that for small edge overlap (equivalently, small O), the error is large, as there is little common structure between the graphs. However, the error rate drops sharply as the edge overlap increases: for 75% and 82% overlap ($O = 54\%$, $O = 66\%$), the error is 13% and 6%, respectively, and for 90% overlap ($O = 82\%$), remarkably almost all nodes are correctly matched. This indicates the applicability of our method for real data even when the underlying graph is not known and the graphs to be matched are not realizations of the edge sampling model.

C. Estimation of Parameters and Priors

As explained in Section VI-B, the matching algorithm requires computing the posteriors which in turn requires the estimation of (i) the prior degree and distance distributions of the underlying graph G, and (ii) the sampling parameters s_1 and s_2 (see Section IV-C). When directly using the edge sampling model (as in Section VI-A), these parameters are known. However, for real graphs, e.g., two snapshots of a network as in Section VI-B, we need to estimate these parameters.

We estimate the underlying degree and distance distributions $p_i(y)$, considering the distributions over the superposition of G_1 and G_2. In other words, we use the empirical degree (or distance) distribution of G_1 and G_2, and compute the underlying degree (or distance) distribution through aggregating the two distributions (i.e.,. adding the probabilities and then normalizing). Although this is a crude approximation, our experiments show that the algorithm works quite well under this approximation.

For the sampling parameters, we can write $e_1 = es_1$ and $e_2 = es_2$, where e, e_1 and e_2 are the expected number of

edges in G, G_1 and G_2, respectively. Also, $e_I = es_1s_2$ corresponds to the expected number of edges that are present in both G_1 and G_2. Using the actual number of edges in G_1, G_2, and in their edge overlap (assuming the number of overlapping edges is known) as an estimation for e_1, e_2 and e_I, we can then estimate s_1 and s_2 using the equations above. This procedure was used in deriving the parameters for the experiments in Section VI-B.

In practice one may not have knowledge (or even an estimation) of the number of edges that overlap in G_1 and G_2, which forbids using the above procedure for estimating s_1 and s_2. An alternative idea is simply to arbitrarily select a large value (e.g., > 0.85) for s_1 (or s_2) to parameterize our method. Preliminary experiments indicate that the method is relatively robust to the choice of parameter s, as long as the *actual* edge overlap is large enough (which is anyways needed for correct matching, as shown in Figure 3(a)). A thorough analysis of this issue is a topic for future work.

D. Matching on Random Graphs

The above results illustrate the performance of our algorithm on real social networks which in turn exhibit (1) a heavy-tailed degree distribution, and (2) a high clustering coefficient. In this section, we discuss the performance of the algorithm on realizations of random graph models where nodes are much more similar to one another, and intuitively much harder to match.

We consider as the underlying fixed graph G a realization of the Erdös-Rényi random graph $G(n,p)$, where n is the number of nodes and each edge is present in the graph independently and with probability p. Such graph is usually sparse and statistically uniform (i.e., nodes tend to "look" similar). This model also yields a binomial, rather than a heavy-tailed, degree distribution for the nodes, and average clustering coefficient of p which is very low compared to social networks of the same size. We then apply the edge sampling process to G in order to obtain G_1 and G_2 and then run our matching algorithm.

For $s = 0.999$ (i.e., very close to 1) we observed via experiments that our algorithm perfectly matches the nodes of $G_{1,2}$, yielding zero error in all experiments. This shows that when the graphs are sufficiently similar, our algorithm is very effective even for large sparse random graphs. On the downside, for $s = 0.95$ (i.e., away from 1), our algorithm cannot match the nodes yielding error rates of over 50%. The problem is that distance is a fragile feature in $G(n,p)$: an edge that is not sampled by the edge sampling process leads to a significant increase in the distance between the two neighbor nodes. In particular, their distance becomes closer to the average distance of the graph. Note that this is significantly different than the model used to capture how distances are elongated in $G_{1,2}$ (see Section IV-C). Moreover, since $G_{1,2}$ are generated independently, distance correlations between pairs of corresponding nodes in the two graphs are also destroyed.

We extend the $G(n,p)$ model to cope with its distance fragility by introducing clustering (i.e., triangles). In particular, for each pair of nodes $u, v \in V$ (where G is a realization of $G(n,p)$) at distance 2, we create an edge (u,v) with probability q. We call this model q-transitive $G(n,p)$ or $G(n,p,q)$. Put it simply, $G(n,p,q)$ models a sparse graph with a non-skewed degree distribution and clustering coefficient controlled by the parameter q. Note that an edge removal in $G(n,p,q)$ with a sufficiently large q will not increase the distance of the incident nodes around its average value.

Figure 3(c) shows the performance of the matching algorithm as a function of the clustering coefficient (controlled by q) when applied to two edge sampled graphs with $s = 0.9$ from an underlying $G(n,p,q)$. The parameter $n = 2000$ and p is chosen such that the average degree remains constant (i.e., 50) in all experiments. Note that each choice of q results in a different clustering coefficient, denoted by C. The results are averaged over different realizations of $G(n,p,q)$. As can be seen in the Figure 3(c)(a), the matching error is a function of the clustering coefficient, and decreases from a high value of 98% when $C \simeq 0.06$ to almost zero when C reaches 0.15. This result indicates that when distances among node pairs are relatively preserved in $G_{1,2}$ (which in this case is a consequence of high clustering) our algorithm can correctly match nodes even when the graphs are sparse and do not exhibit a heavy-tailed degree distribution. For comparison, the EPFL e-mail network has a clustering coefficient of 0.17, and so do many other social networks. However, these also exhibit a heavy-tailed degree distribution which facilitates matching nodes.

Figure 3(b) shows the empirical CCDF of the posterior probability for all n matched pairs for different values of C. Note that for lower clustering, the posteriors are very small (e.g., for $C = 0.06$, only 10% of the matched node pairs have a posterior greater than 0.1) which correlates with the fact that the matching error was very high. However, the posteriors become significantly larger for higher clustering coefficients (e.g., for $C = 0.15$, almost all matched node pairs have a posterior greater than 0.1), again correlating with the fact that the matching error was very low. Thus, as previously observed, the empirical CCDF of the posteriors can indeed be used as a criteria to assess the quality of the matching produced by the algorithm. Moreover, as the clustering increases the performance of $G(n,p,q)$ approaches that of the EPFL dataset, indicating that clustering is an important aspect for correct matching, as it makes pair-wise node distances more robust to random edge removals.

VII. Conclusion

Graph matching has been shown to be very effective at correlating two social networks through their structure. However, most approaches in the literature rely on additional information, either in the form of node or edge attributes, or in the form of a seed set of a priori matched pairs. In particular, the class of methods that build the node map incrementally has been shown to be effective and computationally efficient [12], [18]. However, starting from two

graphs without any side information remains a challenging problem.

We proposed a Bayesian method for approximate graph matching (AGM), based on a generative model that treats the two observable graphs as independent samples of a fixed underlying generator graph. Based on this approach we derive the probability that a mapping between a pair of nodes is correct given their evidence (fingerprints). These probabilities are then used to devise a polynomial-time algorithm where the amount of evidence used to compute the posterior probability of a correct match increases over the iterations of the algorithm by using previously mapped nodes. In a sense, the method tries to match easy pairs first, and then uses these pairs to enrich the fingerprints for the harder pairs. In particular, we apply our method using degree and distances to anchors as evidence. Experiments using real social networks and random graph models indicate that our algorithm can recover from early errors and terminates with an accurate matching (less than 10% error) when the graphs have a reasonable edge overlap (greater than 75%). Finally, we believe our approach can be applied to other domains, particularly when the graphs to be matched can be thought of as noisy observations of a hidden true graph.

ACKNOWLEDGMENT

We are grateful to Arvind Narayanan and Vitaly Shmatikov for sharing their code and data with us, and for their generous help in explaining various aspects of their method and software.

REFERENCES

[1] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore Art Thou R3579X?: Anonymized Social Networks, Hidden Patterns, and Structural Steganography. In *WWW*, 2007.

[2] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola. Learning Graph Matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(6), 2009.

[3] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty Years of Graph Matching in Pattern Recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(03):265–298, 2004.

[4] L. Cordella, P. Foggia, C. Sansone, and M. Vento. A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(10), 2004.

[5] O. Duchenne, F. Bach, I.-S. Kweon, and J. Ponce. A Tensor-Based Algorithm for High-Order Graph Matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(12), 2011.

[6] M. Garey and D. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[7] M. Gori, M. Maggini, and L. Sarti. Exact and Approximate Graph Matching Using Random Walks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(7), 2005.

[8] M. Hay, G. Miklau, D. Jensen, D. Towsley, and C. Li. Resisting Structural Re-Identification in Anonymized Networks. *VLDB Journal*, 19(6), 2010.

[9] B. Luo and E. Hancock. Structural Graph Matching Using the EM Algorithm and Singular Value Decomposition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(10), 2001.

[10] R. Myers, R. Wilson, and E. Hancock. Bayesian Graph Edit Distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(6), 2000.

[11] A. Narayanan, E. Shi, and B. Rubinstein. Link prediction by de-anonymization: How We Won the Kaggle Social Network Challenge. In *IJCNN*, 2011.

[12] A. Narayanan and V. Shmatikov. De-anonymizing Social Networks. *Security and Privacy, IEEE Symposium on*, 2009.

[13] P. Pedarsani and M. Grossglauser. On the Privacy of Anonymized Networks. In *SIGKDD*, 2011.

[14] N. Quadrianto, A. J. Smola, L. Song, and T. Tuytelaars. Kernelized Sorting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(10), 2010.

[15] Y. Tian and J. Patel. TALE: A Tool for Approximate Large Graph Matching. *Data Engineering, International Conference on*, 0:963–972, 2008.

[16] J. Ullmann. An Algorithm for Subgraph Isomorphism. *J. ACM*, 23(1), 1976.

[17] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel. A Practical Attack to De-Anonymize Social Network Users. In *IEEE Symposium on Security & Privacy*, 2010.

[18] L. Yartseva and M. Grossglauser. On the Performance of Percolation Graph Matching. In *ACM Conference on Online Social Networks (COSN)*, 2013.

[19] M. Zaslavskiy, F. R. Bach, and J.-P. Vert. A Path Following Algorithm for the Graph Matching Problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(12), 2009.

[20] S. Zhang, J. Yang, and W. Jin. SAPPER: Subgraph Indexing and Approximate Matching in Large Graphs. *PVLDB*, 3(1):1185–1194, 2010.