On the Performance of Percolation Graph Matching

Lyudmila Yartseva Matthias Grossglauser
School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

firstname.lastname@epfl.ch

ABSTRACT

Graph matching is a generalization of the classic graph isomorphism problem. By using only their structures a graph-matching algorithm finds a map between the vertex sets of two similar graphs. This has applications in the de-anonymization of social and information networks and, more generally, in the merging of structural data from different domains.

One class of graph-matching algorithms starts with a known seed set of matched node pairs. Despite the success of these algorithms in practical applications, their performance has been observed to be very sensitive to the size of the seed set. The lack of a rigorous understanding of parameters and performance makes it difficult to design systems and predict their behavior.

In this paper, we propose and analyze a very simple percolation-based graph matching algorithm that incrementally maps every pair of nodes (i, j) with at least r neighboring mapped pairs. The simplicity of this algorithm makes possible a rigorous analysis that relies on recent advances in bootstrap percolation theory for the $G(n, p)$ random graph. We prove conditions on the model parameters in which percolation graph matching succeeds, and we establish a phase transition in the size of the seed set. We also confirm through experiments that the performance of percolation graph matching is surprisingly good, both for synthetic graphs and real social-network data.

Categories and Subject Descriptors

G.3 [**Mathematics of Computing**]: Probability and Statistics—*Probabilistic algorithms*; G.2.2 [**Discrete Mathematics**]: Graph Theory; H.1 [**Information Systems**]: Models and Principles

Keywords

Graph matching; graph sampling; bootstrap percolation; social networks; de-anonymization

1. INTRODUCTION

Social ties and interactions, interpersonal communications, and information sharing are increasingly conducted through online services and digital media. Both the risks and opportunities this brings are well known and hotly debated. The electronic traces of our patterns of communication, keyword searches, mobility, and information access, give a precise picture of many aspects of our personality and lifestyle. These sources of information are therefore heavily sought after for correlation and mining by advertisers, scientists, governments, and many other entities. In addition, when information from multiple sources and domains is combined, this usually incurs *increasing returns*: the value of a set of databases from different domains (e.g., a social network, a demographic database, and cell phone mobility traces over a given population) is significantly higher than the sum of its values in isolation. It is therefore important to understand how such information gets merged and correlated, and what unintended privacy leaks might result.

One of the most basic representations of such information is a graph, which represents pairwise relationships, potentially enriched with additional attributes. Graphs can describe our phone call patterns, Facebook friendships, relationships in a business organization, or the contact networks that drive the spread of infectious diseases. A typical individual belongs to several such networks, but might possess different identities in different networks, possibly as a deliberate measure to protect her privacy.

In this paper, we study one fundamental aspect of correlating graphs from different application domains: matching their vertex sets through structural information. If the nodes in two graphs use different labels (e.g., phone numbers and e-mail addresses) for two nodes that represent the same underlying entity (e.g., an individual), then we can ask whether the structure of the two graphs reveals the correspondence of some or all of the vertices. In other words, can we find a *matching* between (a subset of) the vertex sets of two graphs? If this is possible, then this has strong implications for privacy and for cross-domain data mining applications.

Recent work answers this question in the affirmative. Narayanan and Shmatikov [10] succeed in matching a large-scale anonymized social network to a second social network that serves as side information. Although the node identities in the first network contain no information per se, the privacy of the network is compromised through the knowledge of a correlated secondary network. In another work, a random graph model served to provide insight on the fundamental feasibility of graph matching for an adversary with unlimited computational power [11]. It turns out that under rather benign conditions, two graphs can be matched perfectly. In summary, this evolving body of work suggests that protecting graph privacy through node anonymization is inadequate.

Graph matching has other important applications in different domains. For example, using graphs to represent images, where vertices are regions and edges are adjacency relations between these regions, is widely applied in recognizing two scenes of an image and in finding similar images. In bioinformatics, modeling gene sequences as graphs and matching these graphs is applied for gene/protein networks alignment [14, 6].

The algorithm in [10] takes as input a *seed set* of prematched node pairs in the two graphs $G_{1,2}$ to be matched. The algorithm then iteratively expands this map, by identifying additional pairs of nodes $(i \in G_1, j \in G_2)$ that can plausibly be matched. Whether (i, j) is a plausible match is computed from the position of i and j with respect to the known mapped nodes. More specifically, they consider nodes i, j in $G_{1,2}$ that are neighbors of at least two mapped nodes, and they propose various heuristics for comparing different candidate pairs. The algorithm has several parameters that need to be tuned through trial-and-error, but it has been shown to perform very well over some real datasets.

Our first key contribution in this paper is a very simple algorithm with a single tuning parameter to perform graph matching. Essentially, the algorithm matches any two nodes $(i \in G_1, j \in G_2)$ that have more than r neighbors already matched to each other. Despite its deceptive simplicity, this algorithm performs very well on real data, and its operation is easy to interpret and control due to a single control nob (r). We believe that this algorithm can be viewed as the canonical placeholder for *percolation graph matching (PGM)* algorithms, which propagate a set of matched node pairs outward. The algorithms in this class differ in how candidate node pairs are compared. As such, PGM should shed light on the qualitative performance of other such algorithms.

This brings us to the second contribution of our paper. A key observation in [10] is the presence of a sharp phase transition in the performance of their algorithm as a function of the seed set size. The algorithm failed almost completely when the seed set size was below a certain threshold, but then shot up to a very high success rate (around 70% in one experiment) when the number of seeds exceeded the threshold. The authors did not speculate as to the reason for the phase transition, or try to characterize it as a function of network properties. In this paper, we formally prove the presence of a phase transition in the seed set size when the input graphs are $G(n, p; s)$. The critical value a_c for the seed set size is a function of the network parameters and of the control parameter r. This result provides a qualitative understanding of this phenomenon and provides quantitative guidelines about the feasible region in the parameter space.

This paper is organized as follows. In Section 2 we give a more in-depth overview of related research. In Section 3, we define a very simple and efficient percolation-based graph matching algorithm. We then describe a stochastic model of this matching algorithm in Section 4. The network model is the $G(n, p; s)$ model introduced in [11]: it generates two correlated Erdös-Rényi random graphs whose similarity can be controlled by a parameter s. In Section 5, we demonstrate the existence of a phase transition in the size of the seed set. When the seed set is smaller than a critical threshold a_c, the algorithm almost certainly fails; if it is larger, the algorithm succeeds in matching almost everything. Our result relies on recent progress in the field of bootstrap percolation [5]. In Section 6, we evaluate the algorithm over both random graphs and real social network data, and we confirm the presence of the phase transition in the seed set. The algorithm also performs remarkably well over real network data. In Section 7 we conclude the paper.

2. RELATED WORK

Graph matching has many applications in several domains, including network de-anonymization, computer vision, databases, and bio-informatics. We discuss the most relevant works here.

In the privacy protection area, there have been several reported successes in matching social network data from different domains [10, 16]. Some of these are computationally expensive and therefore limited to small scales [6]; some rely on assumptions that an attacker is allowed to alter a network before publishing [2] or has access to additional side information, e.g., memberships in groups [16].

To the best of our knowledge, the methods for large-scale attacks proposed in the literature are heuristic in nature [16, 9, 10]. One of the examples of large scale attack is the work of Narayanan and Shmatikov [10]. They were the first to succeed in de-anonymizing two real, large social networks, based only on the network topology. Their approach relies on a seed set of node pairs that are pre-matched, from which they iteratively grow the map.

One key empirical observation in their work is that the size of the seed set is very important: if the seed set is too small, their algorithm tends to quickly die out. The dependence on the size of the seed set seems highly nonlinear, with a sharp transition between almost complete failure and almost complete success in matching as the size of the seed set is increased. This suggests a phase transition phenomenon, which is a major focus of the study of random graphs and percolation models. In this paper, we formally prove a phase transition for a very simple instance of such a percolation graph matching algorithm, thus confirming the empirical observations in their work. Furthermore, we are able to characterize the critical value of the seed set size as a function of graph parameters, including a measure of the similarity of the two graphs to be matched.

Korula and Lattanzi [7] independently propose a graph matching algorithm similar to ours, and they provide an analysis for the $G(n, p; s)$ model as well as for preferential-attachment generator graphs. They consider a regime of dense seeds, where the mapping for a constant fraction of nodes is known a-priori. In this regime, they show that most of the network can be matched with high probability in a single propagation step. Our analysis goes further in that we prove and characterize a phase transition in the size of the seed set. We show that a sublinear seed set size can suffice for matching in some circumstances.

Graph matching also arises in other fields, such as in ontology alignment. Several automated tools were created to match sets of labels describing data [4, 15, 13]. However, the specifics of the problems assume small-scale graphs [4], and the algorithms rely heavily on the properties and attributes of the nodes, rather than on the structural features.

To shed light on the performance of seeded graph matching, we analyze a very simple percolation-based algorithm that incrementally matches pairs of nodes, based on previously matched pairs. In order to make statements about the performance of this algorithm, we need a model for the two graphs $G_{1,2}$ to be matched. For this, we rely on the random graph model introduced in [11]. This model works as follows: we first generate a random graph $G = G(n,p)$ that can be thought of as the true social network. We then derive two observable graphs $G_{1,2}$ from G, by independently including each edge of G in $G_{1,2}$ with probability s. We refer to the this model of a pair of graphs as the $G(n,p;s)$ graph-matching model, where the parameter s controls the similarity (or correlation) between $G_{1,2}$[1].

Our main theoretical contribution in this paper is to identify conditions on the model parameters (n,p,s) and on the size of the seed set a_0 (a small set of initially pre-mapped pairs of nodes) such that percolation graph matching succeeds with high probability. For this, we rely on recent advances in the analysis of bootstrap percolation in the $G(n,p)$ random graph by Janson et al. [5]. We briefly summarize their model and key results here.

Percolation theory is the study of the presence of large (or infinite) clusters in random environments, such as lattices with missing nodes or links, or random graphs. In bootstrap percolation we study systems where a node is part of a cluster only if it has at least r neighbors that belong to the cluster. This more restrictive notion of inclusion can capture, for example, the spread of influence through a social network, where an individual is convinced of an idea only if she hears this idea from several acquaintances.

In a seminal paper [5], Janson et al. succeed in analyzing this process precisely for the Erdös-Rényi ($G(n,p)$) random graph. They stated the following results for $G(n,p)$ infection spread with a threshold r. For given r, n, and p define,

$$t_c := \Big(\frac{(r-1)!}{np^r}\Big)^{1/(r-1)}, \tag{1}$$

$$a_c := \big(1-\frac{1}{r}\big)t_c, \tag{2}$$

$$b_c := n\frac{(pn)^{r-1}}{(r-1)!}e^{-pn}. \tag{3}$$

They analyzed the process and estimated the size of the final active/infected set a^* depending on the size of the initially active set a_0.

THEOREM 1. *[5] Suppose that $r \geq 2$ and $n^{-1} \ll p \ll n^{-1/r}$.*[2]

- *If $a_0/a_c \to \alpha < 1$, then $a^* = (\phi(\alpha) + o(1))t_c$ w.h.p., where $\phi(\alpha)$ is the unique root in $[0,1]$ of*

  $$r\phi(\alpha) - \phi(\alpha)^r = (r-1)\alpha. \tag{4}$$

  [For $r = 2$, $\phi(\alpha) = 1 - \sqrt{1-\alpha}$.]

  Further, $a^*/a_0 \to \phi_1(\alpha) := \frac{r}{r-1}\phi(\alpha)/\alpha$, with $\phi_1(0) := 1$.

- *If $a_0/a_c \geq 1 + \delta$, for some $\delta > 0$, then $a^* = n - o(n)$ w.h.p.; in other words, w.h.p. the process almost percolates. More precisely, $a^* = n - O(b_c)$ w.h.p.*

[1]Note that $G_{1,2} = G(n, ps)$, but their edge sets $E_{1,2}$ are correlated.

[2]In this paper, $f \ll g$ and $f \gg g$ mean $f = o(g)$ and $f = \omega(g)$, respectively.

We use this theorem to analyze the percolation-based matching algorithm in the $G(n,p;s)$ graph model. Although the criterion for propagation is the same in the graph matching process and in the bootstrap percolation ($\geq r$ neighbors infected), the objects of interest in our algorithm are *pairs of nodes* rather than individual nodes as in the Janson et al. model. In other words, in our algorithm, a node pair is mapped if it has at least r neighboring node pairs that are already matched. See the details of the algorithm in the next section.

One key result in the present paper is establishing an equivalence between the percolation process over node pairs for matching and bootstrap percolation, which makes the machinery of [5] available to analyze this process. One subtlety concerns mapping errors: They make the process hard to analyze; and they can propagate, thus reducing the quality of the mapping. To conclude that the algorithm is correct, we need to show two facts: (i) that the matching process percolates and touches "most" nodes, and (ii) that the algorithm matches nodes correctly.

3. PERCOLATION GRAPH MATCHING ALGORITHM

We now describe the graph-matching algorithm, whose analysis is the main contribution of this paper. We are given two graphs G_1 and G_2, both with n nodes[3]. We assume a true but hidden equivalence between nodes in the two graphs, which we can assume w.l.g. to be the identity, with $V_1 = V_2 = V$. The edge sets E_1 and E_2 are in general different, but are correlated. Informally, this means that if an edge (u,v) exists in E_1, it is likely to exist in E_2 as well, and vice versa. The matching algorithm has access only to the structure of the two graphs, i.e., it sees *unlabeled* versions of $G_{1,2}$. Its purpose is to find a map, i.e., a set of tuples $A \subset V_1 \times V_2$ such that each node in $V_{1,2}$ appears in at most one tuple[4]. The map is correct if every element of A is of the form (i,i); for a map with errors, we call $|(i,j) : i \neq j, (i,j) \in A|/|A|$ the *error rate* of the map, and we call $|A|$ the *size* of the map.

Several graph-matching algorithms proposed in the literature assume side information in the form of a known *seed set* A_0 of mapped pairs. These algorithms try to iteratively expand this map by identifying additional pairs of nodes (i,j) in the "vicinity" of the set of confirmed pairs; this process continues until it runs out of pairs to add. Our goal in this paper is to define an algorithm that is simple enough to be tractable, but that also has good matching performance in real scenarios. We refer to our algorithm, and more generally to the class of algorithms that iteratively propagate a map from a seed set, as *percolation graph-matching* (PGM) algorithms, as they rely on a threshold rule reminiscent of bootstrap percolation models [1]. Simply put, a pair (i,j) is added to the set of mapped pairs if there are at least r mapped pairs that are neighbors of (i,j)[5].

We now describe our PGM algorithm more formally. The input of the algorithm is the following:

[3]It is easy to remove the assumption of equal size; its purpose is mainly for notational simplicity.

[4]I.e., A is a matching between the subset of mapped nodes in G_1 and the subset of mapped nodes in G_2.

[5]More precisely, two pairs (i,j) and (i',j') are neighbors iff $(i,i') \in E_1$ and $(j,j') \in E_2$.

- Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$;
- A seed set A_0 of size a_0, consisting of tuples (i, i) of known pairs of matched nodes.

The algorithm we propose and analyze simply maps any two nodes with at least r neighboring pairs already mapped. An equivalent description emphasizes the incremental nature of the process: we associate with every pair of nodes $(i \in V_1, j \in V_2)$ a count of marks $M_{i,j}$. At each time step t, the algorithm *uses* exactly one unused but already mapped pair (i_t, j_t). This pair adds one mark to each neighboring pair, i.e., to every pair in $N_1(i_t) \times N_2(j_t)$. As soon as any pair gets r marks, it is added to the current map; if for some node i there are several nodes j such that all (i, j) have r marks, one pair is picked at random. The process iterates until there are no more unused pairs.

The set $A(t)$ consists of the map built until time t, and the set $Z(t) \subset A(t)$ consists of mapped pairs that have been *used* until t, in the following way:

- At time $t = 0$, $A(0) = A_0$ and $Z(0) = \emptyset$,
- At time step t the algorithm randomly selects a pair $(i_t, j_t) \in A(t-1) \setminus Z(t-1)$ and adds one credit mark to all pairs $(i', j') \in V_1 \times V_2$ such that there exist $(i_t, i') \in E_1$ and $(j_t, j') \in E_2$ (cf. Fig. 1).

  If a pair (i', j') has more than r marks then it is added to the map $A(t)$; furthermore, all other candidates (i'', j') and (i', j'') are permanently removed from consideration.

  Let $\Delta A(t)$ be the set of pairs with r marks, which are added to the map at time t. Then

  $$A(t) = A(t-1) \cup \Delta A(t)$$

  and

  $$Z(t) = Z(t-1) \cup \{(i_t, j_t)\}.$$

  Note that $a(t) \geq z(t) = t$.

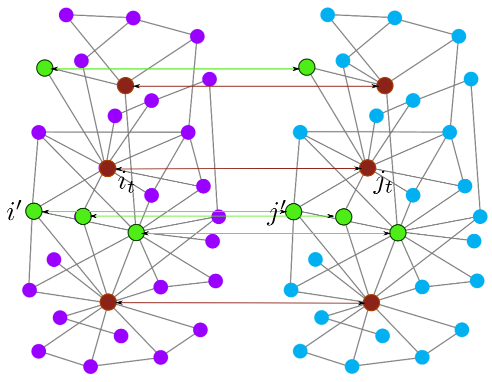

Figure 1: Red nodes are the seeds, green nodes are the set of mapped pairs after the first three iterations, for $r = 2$.

The process stops when $A(t) \setminus Z(t) = \emptyset$, which happens when all pairs from the map $A(t)$ are used. Denote this time step by $T = \min(t \geq 0 \text{ s.t. } A(t) \setminus Z(t) = \emptyset)$. The final map is $A^* = A(T) = Z(T)$ and its size is $a^* = T$.

The role of the parameter r is important: it controls the amount of evidence in favor of a pair of nodes, before these nodes are matched permanently. There is a tradeoff between two types of errors. If r is chosen too low, the probability of a false match increases. If r is chosen too high, then the algorithm may simply run out of candidate pairs to match and stop early.

3.1 Deferred Matching Variant

The algorithm as defined above leads to a tractable probabilistic model, and in particular, can be analyzed using the bootstrap percolation results from [5], as shown below. The basic algorithm greedily matches any candidate pair as soon as it reaches r credits, *even if* $A(t) \setminus Z(t)$ *is not empty.* This is obviously not optimal in most circumstances, as the credits yet to be generated by the remaining pairs in $A(t) \setminus Z(t)$ might improve the credit counts $M_{i,j}$ and avoid matching errors. There is an easy fix to this, which we describe here; we use this variant of the algorithm in the experiments in Section 6.

The modified algorithm works as follows. Whenever $A(t) \setminus Z(t)$ is nonempty, we are conservative and continue to attribute credits to candidate pairs, without forming any new couples. Once $A(t) \setminus Z(t)$ is empty, we form exactly one couple (i, j) that has the maximum $M_{i,j}$ of all candidates (provided this is also above the threshold r; otherwise we stop), and add it to $A(t)$; and so forth.

This variant has the advantage of being conservative about matching new couples: it first uses all the available evidence by using all unused pairs before making irreversible decisions. Also, it makes the choice of the parameter r somewhat less important. In particular, if r is chosen too low, the maximum rule ensures that only the best candidate pairs *relative to other candidates* are matched. Our simulation results show that the variant performs well, but exhibits the same phase transition in r as the basic (greedy) approach.

Formally, at each time-step t,

- The algorithm processes a mapped pair $(i_t, j_t) \in A(t-1) \setminus Z(t-1)$ and adds one credit to every neighboring pair, as in the basic algorithm;
- If $A(t) \setminus Z(t) = \emptyset$, the algorithm takes a pair whose number of credits is maximal and at least r, and adds it to $A(t)$; if there are several such pairs it picks one at random.

The algorithm stops when there are no more pairs with at least r marks.

Our experiments show that this optimization decreases the error rate in certain scenarios, but exhibits similar threshold behavior in the seed set size as the basic version. For more details see Section 6.

4. MODEL

In this section we define the model used for analysis of the PGM algorithm. In the work of Janson [5], the authors proved phase transitions of the size of the final mapping a^* in the initial seed set size a_0. In our model, we show similar phase transitions for PGM.

As we mentioned, we assume the ground-truth network graph G is $G(n, p)$, and two networks are obtained from G as follows: each edge of G is present in the observed network with probability s, independently of everything else. We refer to this probability space as the $G(n,p;s)$ graph matching model [11]. Thus an input of the problem is the following:

- Two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, obtained as a realization of the $G(n,p;s)$ graph matching model;
- A seed set $A_0 \subset V \times V$ of size $a_0 = |A_0|$.

Our goal is to explore the following question: Under what conditions on the model parameters n, p, s, a_0 and r does the algorithm propagate and match the two graphs (almost) correctly?

4.1 Properties of the Propagation Process

Let $E(i,i')$ denote the event that the edge (i,i') is present in G; and $E_1(i,i')$ and $E_2(j,j')$ are the events that edges (i,i') and (j,j') occur in G_1, G_2, respectively.

OBSERVATION 1. *Since the graph G is $G(n,p)$, the unconditional edge probability $P\{E_1(i,i')\} = P\{E_2(j,j')\} = ps$. Butsince G_1 and G_2 are sampled from the same generator,*

$$P\{E_1(i,i')|E_2(i,i')\} = s.$$

For convenience of notation, we omit the reference to the graph when it is clear from the context, and we refer to the nodes of G_1 by index i and to nodes of G_2 by index j. We write $E_1(i,i_t)$ as $E_{i,t}$ and $E_2(j,j_t)$ as $E_{j,t}$. $i = j$ means that i and j correspond to the same node of G.

Let $I_{i,j}(t)$ be an indicator of the event that a pair (i,j) received a mark at time step t, as a result of using a pair (i_t, j_t). This is equivalent to the event that there exist edges $(i,i_t) \in G_1$ and $(j,j_t) \in G_2$. Hence its probability is

$$\mathsf{P}\{I_{i,j}(t)=1\} = \mathsf{P}\{E_{i,t}, E_{j,t}\}.$$

We state the following lemma about the increments at time t, conditional on no matching errors so far:

LEMMA 1. *Conditional on $i_\tau = j_\tau$ for all $\tau \le t$*

1. $P\{I_{i,j}(t) = 1\} = \begin{cases} (ps)^2, i \neq j \\ ps^2, i = j \end{cases}$
2. *For a fixed t, the $\{I_{i,j}(t)\}_{i,j,i\neq j}$ are not independent.*
3. *For a fixed t, the $\{I_{i,i}(t)\}_i$ are independent.*
4. *For fixed $t_1 \neq t_2$ and i, j, the $I_{i,j}(t_1)$ and $I_{i,j}(t_2)$ are independent.*

PROOF. Conditional on $i_\tau = j_\tau$ for all $\tau \le t$

1. If at time t, a seed is mapped correctly, the nodes i_t and j_t are sampled from the same node of G, then by Observation 1,

$$\mathsf{P}\{I_{i,j}(t) = 1\} = \mathsf{P}\{E_{i,t}, E_{j,t}\} = \begin{cases} (ps)^2, i \neq j \\ ps^2, i = j \end{cases}$$

2. For $i \neq j$ and $i_1 \neq j$:

$$\begin{aligned} &\mathsf{P}\{I_{i,j}(t)=1|I_{i_1,j}(t)=1\} = \\ &= \mathsf{P}\{E_{i,t}, E_{j,t}|E_{i_1,t}, E_{j,t}\} \\ &= \mathsf{P}\{E_{i,t}|E_{i_1,t}\} = \mathsf{P}\{E_{i,t}\} = ps \end{aligned}$$

3. For $i = j$ and $i_1 = j_1$ $(i_1 \neq i)$:

$$\begin{aligned} &\mathsf{P}\{I_{i,j}(t)=1|I_{i_1,j_1}(t)=1\} = \\ &= \mathsf{P}\{E_{i,t}, E_{j,t}|E_{i_1,t}, E_{j_1,t}\} \\ &= \mathsf{P}\{E_{i,t}, E_{j,t}\} = (ps)^2 \end{aligned}$$

4. For $t_1 \neq t_2$:

$$\begin{aligned} &\mathsf{P}\{I_{i,j}(t_1)=1|I_{i,j}(t_2)=1\} = \\ &= \mathsf{P}\{E_{i,t_1}, E_{j,t_1}|E_{i,t_2}, E_{j,t_2}\} \\ &= \mathsf{P}\{I_{i,j}(t_1) = 1\} = (ps)^2 \end{aligned}$$

□

Clause 2 states that markers obtained for two different pairs with a node in common are not independent. Given that a pair (i,j) gets a mark, the event that another pair (i_1, j) also gets a mark is more likely. Clause 3 is key for further analysis of the process. It states that correctly mapped pairs obtain marks independently. Thus, if a pair (i,i) got a mark at time t, it does not correlate with (j,j) getting a mark. Clause 4 asserts that each seed spreads its marks independently. In other words, at a time step t, a pair gets a mark independently of other time steps.

The count $M_{i,j}(t)$ is the number of marks of (i,j) at time t:

$$M_{i,j}(t) = \sum_{s=1}^{t} I_{i,j}(s).$$

Under the conditions of Lemma 1, each $M_{i,j}(t)$ is the sum of i.i.d. Bernoulli random variables, so it is either a $\mathsf{Bi}\left(n,(ps)^2\right)$ for $i \neq j$ or a $\mathsf{Bi}\left(n, ps^2\right)$ for $i = j$. In the following section, we develop conditions when PGM does not match wrong pairs (w.h.p.).

5. PERFORMANCE OF PGM

5.1 Main Theorems

Let $q = ps^2$ and $r \ge 2$, and note that q is the probability of an edge being sampled in both G_1 and G_2 or, equivalently, the probability of an edge to be contained in the intersection of the edge sets $E_1 \cap E_2$. Define

$$t_c = \Big(\frac{(r-1)!}{nq^r}\Big)^{\frac{1}{r-1}} \text{and } a_c = \big(1 - \frac{1}{r}\big)t_c. \qquad (5)$$

Here we show that t_c and a_c are the critical time horizon and the critical value of the initial size of the seed set, respectively. This means that for an initial number of seeds a_0 lower than a_c, the PGM algorithm stops earlier than t_c, with the final size at most $2a_0$; for a_0 larger than a_c, the algorithm propagates to most of the graph.

THEOREM 2 (SUBCRITICAL REGIME). *For $n^{-1} \ll ps^2 \ll sn^{-\frac{3}{2r}}$ if $a_0/a_c \to \alpha < 1$, the propagation algorithm stops with $a^* \le t_c$ w.h.p. In particular $a^* = (\phi(\alpha) + o(1))t_c \le \frac{r}{r-1}a_0$, where $\phi(\alpha)$ is the unique root in $[0,1]$ of $r\phi(\alpha)) - \phi(\alpha)^r = (r-1)\alpha$.*

This means that in the subcritical regime, the final map is only slightly larger than the seed set, because the mapping process does not percolate. Now we consider at what happens above the threshold $a_0 > a_c$.

THEOREM 3 (SUPERCRITICAL REGIME). *For $n^{-1} \ll ps^2 \ll sn^{-\frac{3}{2r}}$, if $a_0/a_c \geq 1+\delta$ the algorithm propagates, and the size of the final mapping is $a^* = n - o(n)$ w.h.p.*

In summary, there is a sharp phase transition at $a_0 = a_c$ that separates almost certain failure from almost certain success of the percolation graph matching process. We discuss the implications of this phase transition and the scaling of the main parameters in more detail in Subsection 5.4.

5.2 Proof Sketch and Bootstrap Percolation

We briefly outline the main steps of the proof and provide full details in the next subsection.

Our main goal is to prove that the couple formation process $A(t)$ defined in the previous section can be analyzed using the bootstrap percolation model introduced in [5]. In summary, [5] analyses a process where, at every time step t, objects collect a credit with probability p, independently of everything else. We want to analyze the PGM algorithm within the $G(n,p;s)$ model. However, our object of interest is not an individual node, but a pair (i,j).

At every time-step, one pair spreads credits to other node pairs. However, we do not have the critical feature that makes the analysis of [5] tractable: as shown in Lemma 1, the credit increments $I_{i,j}(t)$ are not equiprobable, and they are not independent. Therefore, the results of [5] cannot be applied directly.

Fortunately, the specific structure of the process of increments over pairs reveals a way out. The key observation is that the credits of *correct* pairs $M_{i,i}(t)$ are in fact independent of each other. Another observation of Lemma 1 is that correct pairs are more likely to get a mark. Thus, the (small) subset of pairs of the form (i,i) within all the possible pairs $V \times V$ can be analyzed using the bootstrap percolation framework.

Therefore, we first consider the event X that at any time t, a wrong pair (i,j) has collected at least r credits without either of the "competing" correct pairs (i,i) and (j,j) having collected r credits. In the case of event X, it is possible (but not guaranteed) that a matching error has occurred. In Lemma 2 below, we show that $\mathsf{P}\{X\} \to 0$ under appropriate conditions. Under these conditions, the matching algorithm does not make wrong matches (w.h.p.) and is suitable for further analysis.

It remains to be shown whether the algorithm percolates. For this, it is conservative to only consider the credits $M_{i,i}$ attributed to correct pairs, as the probability to percolate can only increase by adding additional pairs into the system. As the correct counts are independent binomials, it is then straightforward to map the problem into the bootstrap percolation framework.

5.3 Proofs of Theorems 2 and 3

In this section, we use the results from [5] to formulate our key results. We show a sharp face transitions in the final map size a depending on $a_0 < a_c$ or $a_0 > a_c$.

A key lemma bounds the probability that no error happens in the matching process. An error may occur if at some time step, a bad pair (i,j) collects r marks before its adjacent good pairs (i,i) and (j,j) have collected more than r marks. If such errors are very rare, then we can focus only on correctly mapped pairs in the analysis of $A(t)$. Let $X_{i,j}(t)$ denote the event that the algorithm made an error at time step t by mapping a pair $(i,j), i \neq j$, where $r \leq t \leq n$. The probability of this event is

$$\begin{aligned} &\mathsf{P}\{X_{i,j}(t)\} \leq \\ &\leq \mathsf{P}\{M_{i,j}(t) = r, M_{i,i}(t) \leq r, M_{j,j}(t) \leq r\} \end{aligned}$$

Denote by $X = \bigcup_{t, i\neq j} X_{i,j}(t)$ an event that at any time-step t an error happened.

LEMMA 2. *If $ps \ll n^{-\frac{3}{2r}}$ (with $2 \leq r \leq n$), then $\mathsf{P}\{X\} \to 0$ with $n \to \infty$,*

The proof of the lemma is in the appendix.

Therefore for $ps \ll n^{-\frac{3}{2r}}$, Lemma 2 guarantees that w.h.p. we need only consider the evolution of the correct counts $\{M_{i,i}\}$, to which we can apply the results of [5] directly.

PROOF THEOREM 2 AND 3. The PGM process restricted to correct pairs (i,i) is isomorphic to the bootstrap percolation process for a $G(n,q)$ random graph, with $q = ps^2$. Consider the two events $\{\{M_{i,i}\}$ percolates $\}$ and $\overline{X}$. In the supercritical case, by virtue of Theorem 1 and Lemma 2, both events occur with high probability. Therefore, PGM percolates correctly and to a set of size $n - o(n)$ w.h.p.

In the subcritical case, the process $\{M_{i,i}\}$ does not percolate. As $\mathsf{P}\{X\} \to 0$, the full PGM process over all pairs does not percolate either by virtue of Lemma 2. □

5.4 Interpretation of Results

Here we look into more details on the parameters of the algorithm. In particular, we consider how the threshold a_c scales with respect to r, p and s, and we elaborate on what happens near the bounding conditions on $q = ps^2$.

The parameter r controls a tradeoff between matching errors and percolation blocking. If r is too low, then a wrong pair (i,j) might accumulate r credits before the correct pairs (i,i) and (j,j) do; if r is too high, the process might not percolate, and most nodes do not get matched. Note that r has to be at least 2 for the algorithm to work: for $r = 1$, the algorithm would match pairs of nodes with only one mapped neighbor, which would necessarily lead to ambiguity, except in degenerate cases.

The lower bound $\frac{1}{n} \ll q$ simply ensures that the intersection of the two graphs has a giant component, without which the algorithm cannot percolate. The upper bound $q \ll sn^{-\frac{3}{2r}}$ is more subtle. Of course, if q exceeds the upper bound, the algorithm still percolates, but it will make errors. This is because the ratio in the probabilities of generating correct and wrong credits is not large enough to guarantee $\overline{X}$. As expected, the threshold a_c is decreasing with increasing p and s, so denser graphs require smaller seed sets. For most scenarios of practical interest, r would be a constant. For example, if s is a constant, and the mean degree np grows sub-linearly, then there is a constant r that satisfies the upper bound $q \ll sn^{-\frac{3}{2r}}$. Specifically, if we scale $nq = n^{\delta}$, with $0 < \delta < 1$ a constant, then the seed set threshold then scales as $a_c \propto n^{1-\delta\frac{r}{r-1}}$. For densification slower than a power law, suppose the mean is $nq = \Theta(\log n)$ (which is the threshold for the disappearance of symmetry and of isolated vertices [3]), then a_c scales as follows: $a_c = (1-\frac{1}{r})(r-1)!^{\frac{1}{r-1}} n(\log n)^{-r/r-1}$. With $r = 2$ this is $a_c = \frac{n}{\log^2 n}$.

6. SIMULATION RESULTS

In this section, we test the PGM algorithm over real and artificial graphs, with two goals: to validate the phase transitions predicted by theory, and to check how well the algorithm performs on real networks.

To evaluate the performance of the algorithm, we use two metrics: The first is the size of final the map a^* (the total number of mapped nodes), which says how far the algorithm propagates. The second is the error rate, i.e., the fraction of wrong pairs in the map. Recall that the error rate is $\frac{|(i,j):i\neq j,(i,j)\in A^*|}{a^*}$.

The following ground-truth network graphs are considered:

- Erdös-Rényi random graph $G(n,p)$;
- Slashdot social network;
- EPFL e-mail exchange network;
- Geometric random graph $G_{geom}(n,d)$.

We run the deferred matching version of the algorithm (see Section 3.1) with $r = 2$; however the results are qualitatively similar for the basic version. For the $G(n,p)$, Slashdot and $G_{geom}(n,d)$ random graph, we use the edge sampling model: each edge appears in the observed network with probability s. The experiment with the EPFL network is in some sense more challenging, because the two networks to be matched are in fact different observations of the social interactions within an organization at two different points in time. Figures 2 - 8 show the dependence of the performance metrics on the size of the seed set a_0. Each figure contains 3 curves for different values of the graph similarity parameter, which is either the sampling parameter s, or an estimate of s in the case of the EPFL dataset. The parameter s determines the size of the overlap of the observed networks: the intersection of the edge sets of the two graphs are of size proportional to s^2. We averaged all the results over 10 realizations.

6.1 $G(n,p;s)$ Model

To support our results, we first simulate the $G(n,p;s)$ graph matching model exactly. Specifically, in this model, the generator graph G is an Erdös-Rényi $G(n,p)$ graph with $n = 50000$ and $p = 20/n$.

We observe that when the size of the seed set is sufficiently large, the algorithm propagates to the complete mapping (see Figure 2). We also see the sharp phase transitions predicted in Theorems 2 and 3. Furthermore, the theoretically obtained threshold a_c appears very precise. According to the definition (5) of a_c, for the first curve, $s = 0.9$ the critical size of the seed set a_c is 96, for $s = 0.8$ the a_c is 152 and for $s = 0.7$ the a_c is 260. We can see that the observed transitions are close to these values. To highlight this fact, we normalize the x-axis by a_c. In Figure 3, we observe that, after re-scaling, all the curves look essentially the same.

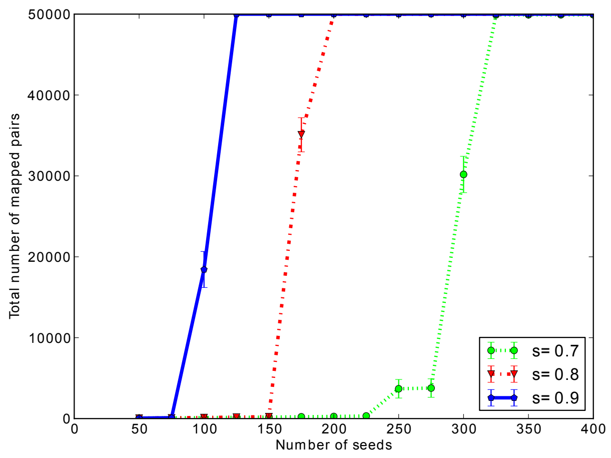

Figure 2: Total number of mapped nodes vs number of seeds for the PGM algorithm over $G(n,p)$ with $n = 50000$ and $p = 20/n$.

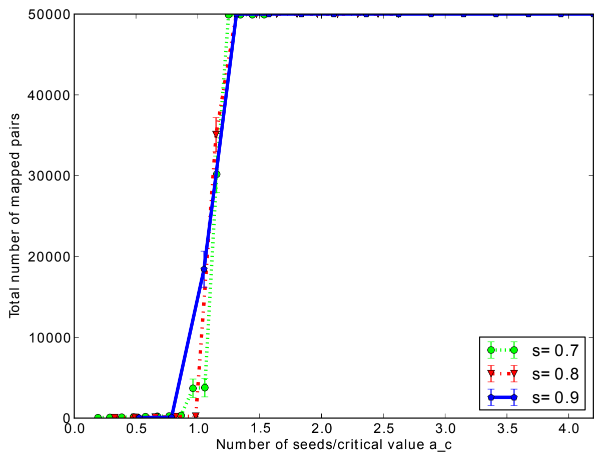

Figure 3: Total number of mapped nodes vs number of seeds for the PGM algorithm over $G(n,p)$ with $n = 50000$ and $p = 20/n$. The x-axis is rescaled according

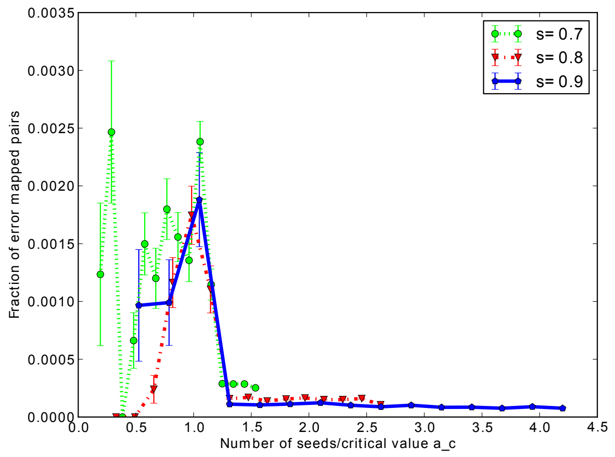

Figure 4: Error rate vs number of seeds for the PGM algorithm over $G(n,p)$ with $n = 50000$ and $p = 20/n$.

The Figure 4 provides further detail about the behavior of the PGM algorithm: if the size of the seed set a_0 is very small the PGM dies out quickly, adding only a small number of very unreliable matches to the seed set. This results in a lot of noise in the subcritical regime. For very large $a_0 \gg a_c$, the error rate goes to zero, thanks to the increasing amount of side information, as expected by Lemma 2. There appears to be a peak in the error rate just at the onset of percolation, because the algorithm matches almost the whole network, but with barely enough seeds to percolate. Errors are then more likely than for larger seed sets.

To confirm that deferred version does not change the observed phenomenons, we also ran the analogous experiments with the basic version of the PGM and observe identical threshold behavior.

6.2 Real Networks: Slashdot and E-mail Graphs

In the second set of experiments, we run PGM over large-scale social networks. First, we run the algorithm over real friend/foe links between Slashdot users [8] obtained in November 2008 (cf. Table 1).

Nodes	77360
Edges	905468
Number of components	1
Average clustering coefficient	0.0555
Diameter (longest shortest path)	10

Table 1: Slashdot dataset statistics.

To generate two observations of the network, we resort to edge sampling. In this model, when $s = 0.9$, the overlap of the two networks is less than 63000 nodes; when $s = 0.8$, the overlap is about 49000 nodes; when $s = 0.7$, it is 38000 nodes.

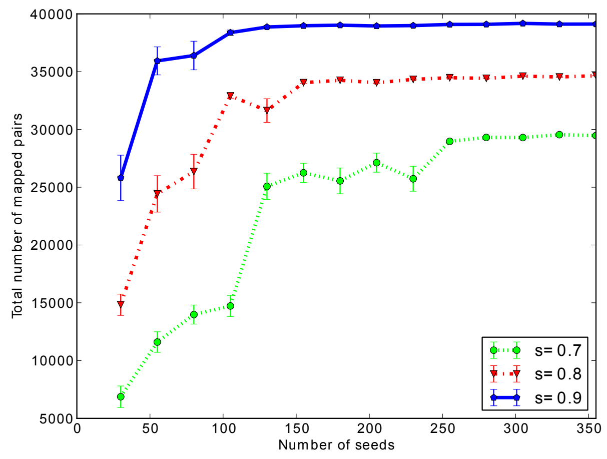

Figure 5: Total number of mapped nodes vs number of seeds for the PGM algorithm over the Slashdot network.

The results suggest phase transitions in the size of the final mapping, albeit less sharp than for $G(n; p)$ (see Figure 5). We also see that if the algorithm propagates (supercritical case), the error rate is encouragingly small (see Figure 6). For example, for $s = 0.9$, it is enough to have 150 seeds (which is 0.2% of all nodes) for the algorithm to propagate over the majority of the graph. Figure 6 shows that the error rate drops rapidly with a_0.

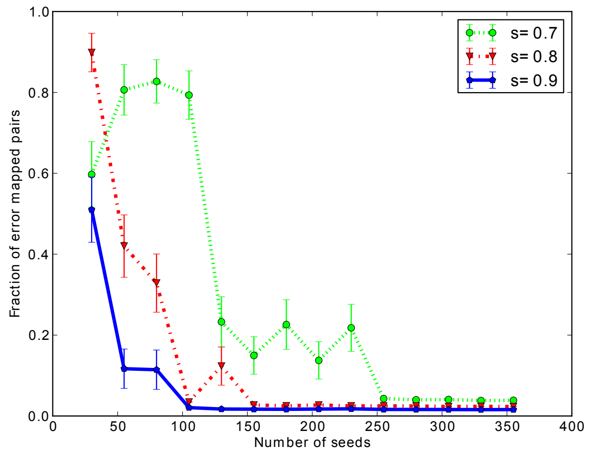

Figure 6: Error rate vs number of seeds for the PGM algorithm over the Slashdot network.

Second, we obtained snapshots of the e-mail traffic on the EPFL campus for different time periods (the week numbering starts at the beginning of year). Each node corresponds to an e-mail account, and an undirected edge means that at least one e-mail was sent between two accounts. The experiment is more realistic in the sense that we do not rely on the sampling model to generate two similar graphs $G_{1,2}$, but instead these graphs correspond to the real traffic patterns in two different time periods.

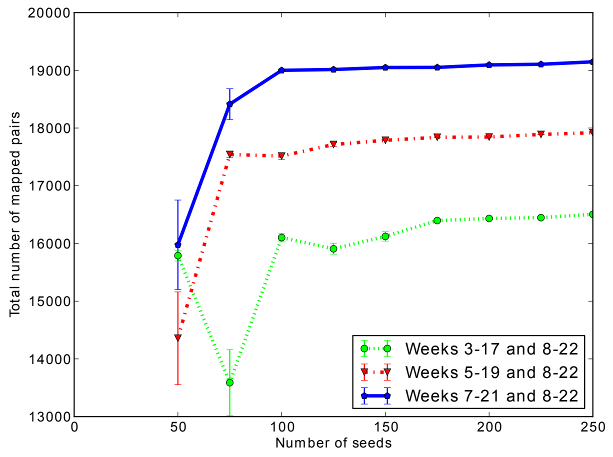

Figure 7: Total number of mapped nodes vs number of seeds for the PGM algorithm over the EPFL contact network.

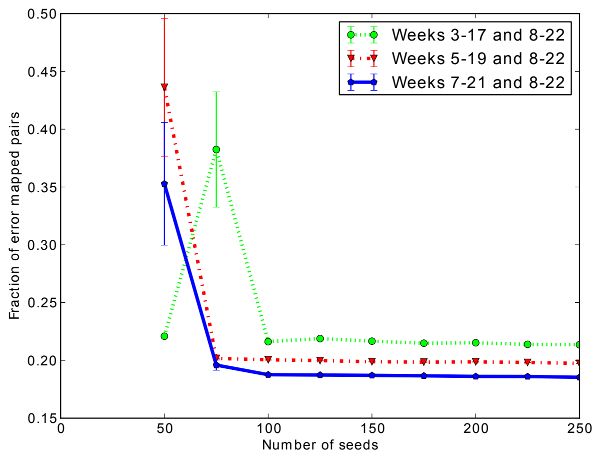

Figure 8: Error rate vs number of seeds for the PGM algorithm over the EPFL contact network.

The challenge for the algorithm is that in the considered graphs not only edge sets are different, but so are vertex sets. In other words, the PGM does not match the vertex sets of two graphs anymore, instead it identifies common subsets and matches them. If the two graphs are different enough the PGM can not separate the nodes which are not presented in both graphs and tries to match them thus increasing the error rate. Another challenge is that graphs are quite sparse, the average degree is about 7. The three curves demonstrate the behavior of the algorithm on the e-mail exchanges graphs for the following periods:

- G_1 is a graph of e-mails sent between weeks 3 and 17 and G_2 is a graph e-mails sent between weeks 8 and 12. Each graph contains approximately 60 000 nodes and 230 000 edges. The intersection graph has 50000 nodes and 160000 edges.
- For weeks 5-19 and 8-12, respectively: Each graph contains approximately 61 500 nodes and 231 000 edges. The intersection graph has 54000 nodes and 185000 edges.
- For weeks 7-21 and 8-12, respectively: Each graph contains approximately 61 500 nodes and 231 000 edges. The intersection graph has 59000 nodes and 207000 edges.

The results reveal similar phase transitions on the size of the final mapping and error rate as for those in $G(n; p)$ and Slashdot(see Figures 7 and 8).

6.3 Random Geometric Graph $G_{geom}(n, d)$

The performance of our proposed PGM algorithm is surprisingly good over both the $G(n, p; s)$ random graphs and over real social networks. We conjecture, however, that its success relies in part on the compactness of these graphs, which ensures that even with a relatively small number of seeds, every node in the network is close to some seeds, which allows to "triangulate" the nodes.

To illustrate this, we report on an experiment where the generator graph G is a random geometric graph $G_{geom}(n, d)$. A random geometric graph is a random undirected graph which is generated by placing vertices uniformly at random on the unit square $[0, 1)^2$. Two vertices u and v are connected if and only if the distance between them is at most d [12]. The typical distance in a supercritical random geometric graph scales as $n^{1/2}$, in contrast to the logarithmic distance in $G(n, p)$ and other "small-world" networks. The average degree of a geometric graph is πnd^2. For our settings $n = 30000$ and $d = 0.01$, for an average degree of approx. 10.

Figures 9 and 10 show the experiment for $G_{geom}(n = 30000, d = 0.01)$. We observe that the algorithm does not percolate, and that it has a very high error rate within the map. While a complete understanding of the limits of percolation-based graph matching is lacking, this does suggest that PGM performs better with compact networks.

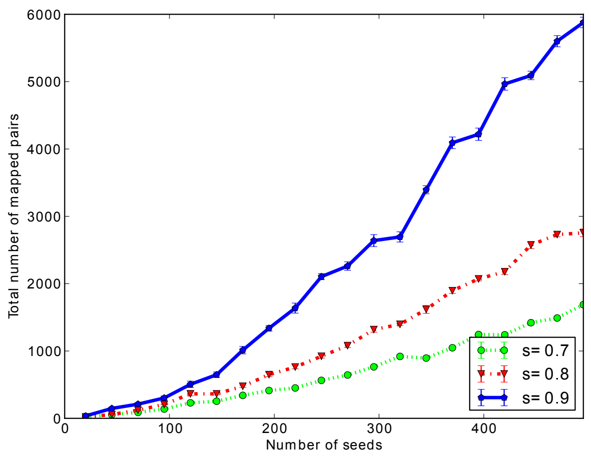

Figure 9: Total number of mapped nodes vs number of seeds for the PGM algorithm over $G(n, d)$ random geometric graph model where $n = 30000$ and $d = 0.01$.

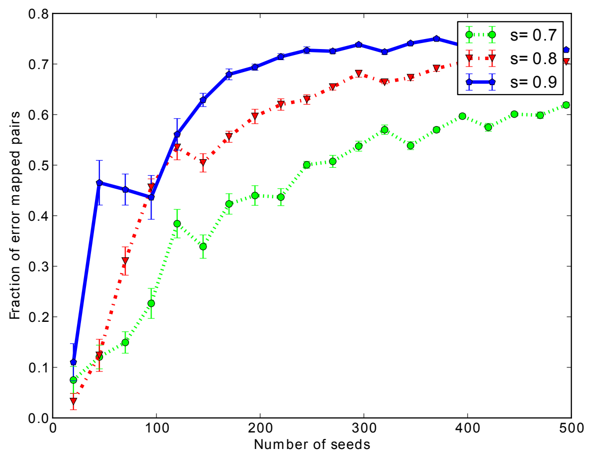

Figure 10: Error rate vs number of seeds for the PGM algorithm over $G(n, d)$ random geometric graph model where $n = 30000$ and $d = 0.01$.

7. CONCLUSION

We developed an algorithm for large-scale graph matching, and proposed a theoretical framework for its analysis. This enabled us to investigate conditions on the graph and algorithm parameters such that the algorithm percolates and performs well. One of our key contributions is a theoretical model and the identification of a phase transition in the size of the seed set, i.e., side information provided to the algorithm.

We have run experiments on several types of networks, confirming the results. We have observed two key phenomena:

- There is a sharp phase transition of the size of the final map a^*, depending on the size of the initial seed set.

- The algorithm has low complexity and performs well on real and artificial compact graphs; however, its performance was much worse on the random geometric graph, which is less compact.

The choice of r can have a significant impact on the map propagation process. A larger r means that we insist on more evidence before permanently mapping two nodes. While Lemma 2 suggests a rather benign condition on r, in more challenging scenarios (e.g., if the vertex sets V_1 and V_2 overlap only partially, or when some seeds are wrong), a larger threshold r can help compensate for such vagaries. We have also run experiments to check how robust the algorithm is with respect to these perturbations and observed that seed errors affect mostly the onset of percolation, but with a minor effect on the error rate. This agrees with the arguments in Lemma 2. An interesting related question concerns the choice of good seeds in situations where we are afforded some control.

In summary, we believe that the percolation-based graph matching algorithm proposed in this paper is both practically relevant, given its good performance and low complexity, and theoretically instructive, given the precise statements we can make about the required size of the seed set and the percolation dynamics. We hope that this work opens up new avenues for research, both into better graph-matching algorithms and richer classes of network models.

APPENDIX

Proof of Lemma 2

Proof. First we bound the probability of mapping a wrong pair (i, j) $(i \neq j)$ at time step t, conditional on no wrong used pairs up to time $t-1$. Note that conditioning on a correct used pair $i_\tau = j_\tau$ at time τ implies that this pair was correctly matched at some time τ' before τ, which in turn ascertains that for this pair, the correct count $M_{i_\tau,i_\tau}(\tau')$ "won" over all wrong counts $M_{i',i_\tau}(\tau')$ and $M_{i_\tau,j'}(\tau')$. Therefore, conditional on $i_\tau = j_\tau$ for $\tau < t$, correct counts are stochastically (slightly) larger, and wrong counts stochastically smaller. In the following argument, we are conservative in ignoring this bias in bounding the probability of future errors.

$$\begin{aligned}
&\mathsf{P}\{X_{i,j}(t)\} \leq \\
&\qquad \leq \mathsf{P}\{M_{i,j}(t) = r, M_{i,i}(t) \leq r, M_{j,j}(t) \leq r\} \\
&\qquad \leq \mathsf{P}\{M_{i,j}(t) = r, M_{i,i}(t) \leq r\}
\end{aligned}$$

We compute this probability as follows (see Figure 11 for illustration).

1. Let L be the subset of used nodes $i_1, \ldots, i_t$ in G_1 up to time t that have an edge to i. Define $l = |L|$.

2. To have $\{M_{i,j}(t) = r\}$, there need to be r edges from used nodes $j_1, \ldots, j_t$ in G_2 to node j. This ensures that (i, j) has accumulated exactly r credits from $(i_1, j_1), \ldots, (i_t, j_t)$.

3. To have $\{M_{i,i}(t) \leq r\}$, out of the set L at most r can be connected to i in G_2.

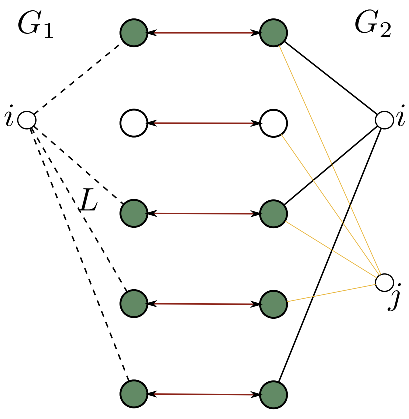

Figure 11: Edges to used nodes for pairs (i, j) and (i, i).

Thus,

$$\begin{aligned}
&\mathsf{P}\{X_{i,j}(t)\} \leq \\
&\leq \sum_{r<|L|=l<t} \mathsf{P}\{|L| = l\}\, \mathsf{P}\left\{M_{i,j}(t) = r, M_{i,i}(t) \leq r \middle| |L| = l\right\} \\
&\leq \sum_{r<l<t} \mathsf{P}\{\mathsf{Bi}(t, ps) = l\}\, \mathsf{P}\{\mathsf{Bi}(l, ps) = r\}\, \mathsf{P}\{\mathsf{Bi}(l, s) \leq r\},
\end{aligned}$$

where the latter factors because the set of edges determining $M_{i,i}$ and $M_{i,j}$ are disjoint. Now we compute this probability explicitly.

$$\begin{aligned}
&\mathsf{P}\{X_{i,j}(t)\} \leq \\
\leq\ & \sum_l \frac{t!}{l!(t-l)!} \frac{l!}{r!(l-r)!} (ps)^l (1-ps)^{t-l} (ps)^r (1-ps)^{l-r} \cdot \\
& \cdot \mathsf{P}\{\mathsf{Bi}(l, s) \leq r\} \\
\overset{(a)}{\leq}\ & \mathsf{P}\{\mathsf{Bi}(t, ps) = r\}\, (t-r)!\, e^r \sum_l \frac{(ps)^l \exp(-ls/2)}{(t-l)!(l-r)!} \\
\overset{(b)}{\simeq}\ & \mathsf{P}\{\mathsf{Bi}(t, ps) = r\}\, (t-r)^{t-r} e^r \sum_l \frac{(ps)^l \exp(-ls/2)}{(t-l)^{t-l}(l-r)^{l-r}}, \qquad (6)
\end{aligned}$$

where (b) uses the Stirling formula for factorials, and (a) uses the following Chernoff bound for the left tail of the binomial:

$$\mathsf{P}\{X < (1-\sigma)\mu\} \le \exp\big(\frac{-\sigma^2\mu}{2}\big).$$

Here X is $\mathsf{Bi}(l, s)$, $\mu = ls$ and $\sigma = 1 - \frac{r}{ls}$ (to make $(1-\sigma)\mu = r$). Then,

$$\begin{aligned}\mathsf{P}\{\mathsf{Bi}(l,s) \le r\} &\le \exp\left(\frac{-(1-\frac{r}{ls})^2 ls}{2}\right)\\ &= \exp\big(-\frac{ls}{2} + r - \frac{r^2}{2ls}\big)\\ &\le \exp(r - ls/2)\end{aligned}$$

Now we denote by $g(l) = \frac{(ps)^l \exp(-ls/2)}{(t-l)^{t-l}(l-r)^{l-r}}$ the terms in the above sum.

We upper-bound $\sum_{r<l<t} g(l)$ in (6) by $(t-r)g(l_0)$ by virtue of Lemma 3 below:

$$\begin{aligned}g(l_0) =& \frac{x^{tx+r}}{(t(1-x)+r)^{t(1-x)+r}(tx)^{tx}}\\ =& \frac{x^r}{(t(1-x)+r)^{t(1-x)+r}(t)^{tx}}\\ \simeq& \frac{x^r}{(t+r)^{t+r}t^{tx}}\end{aligned}$$

Finally we upper-bound the error probability $\mathsf{P}\{X_{i,j}(t)\}$ as follows:

$$\begin{aligned}\mathsf{P}\{X_{i,j}(t)\} \le&\\ \le& \mathsf{P}\{\mathsf{Bi}(t, ps) = r\}(t-r)^{t-r+1}e^r g(l_0)\\ \le& t^r(ps)^r(t-r)^{t-r+1}e^r g(l_0)\\ \le& (ps)^r t^{t+1} e^r g(l_0)\\ \simeq& (psq)^r e^r \frac{t^{t+1}}{(t+r)^{t+r}t^{tq}}\\ \le& \frac{(psq)^r}{t^{tq}}\\ \le& (ps)^{2r}\end{aligned}$$

Now to estimate a total probability of an error X, we take a union bound for all i, j, t. Thus we get $P(X) \le n^3(ps)^{2r}$, which provides a condition $ps \ll n^{-\frac{3}{2r}}$. □

Lemma 3. *For $x = pse^{-s/2} = o(1)$ the function $g(l)$ reaches its maximum at $l_0 = tx + r$.*

Proof.

$$g(l) = \exp\big(l\log x - (t-l)\log(t-l) - (l-r)\log(l-r)\big)$$

We find a maximum of the function $\hat{g}(l)$, where $g(l) = \exp \hat{g}(l)$

$$\begin{aligned}\hat{g}'(l) =& \log x + \log(t-l) + \frac{t-l}{t-l} - \log(l-r) - \frac{l-r}{l-r}\\ =& \log\Big(\frac{(t-l)x}{l-r}\Big)\end{aligned}$$

To find an extremum, we solve $\hat{g}'(l) = 0$ or $\frac{(t-l)x}{l-r} = 1$. We get $l_0 = \frac{tx+r}{1+x} \simeq tx + r$ and it is easy to check that this is a local maximum. □

Acknowledgements

The authors are grateful to Augustin Chaintreau for valuable feedback and constructive criticism.

A. REFERENCES

[1] M. Aizenman and J. L. Lebowitz. Metastability effects in bootstrap percolation. *Journal of Physics A: Mathematical and General*, 21(19), 1988.

[2] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, 2007.

[3] B. Bollobás. *Random graphs*, volume 73. Cambridge University Press, 2001.

[4] P. Doshi, R. Kolli, and C. Thomas. Inexact matching of ontology graphs using expectation-maximization. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(2), 2009.

[5] S. Janson, T. Łuczak, T. Turova, and T. Vallier. Bootstrap percolation on the random graph $G_{n,p}$. *The Annals of Applied Probability*, 22(5), 2012.

[6] G. W. Klau. A new graph-based method for pairwise global network alignment. *BMC Bioinformatics*, 10(S-1), 2009.

[7] N. Korula and S. Lattanzi. An efficient reconciliation algorithm for social networks. *ArXiv e-prints*, July 2013.

[8] J. Leskovec. Stanford network analysis project. `http://snap.stanford.edu/index.html`.

[9] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are who you know: inferring user profiles in online social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM, pages 251–260, 2010.

[10] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, 2009.

[11] P. Pedarsani and M. Grossglauser. On the privacy of anonymized networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD, 2011.

[12] M. Penrose. *Random Geometric Graphs.* Oxford University Press, USA, 2003.

[13] M.-E. Rosoiu, C. T. dos Santos, and J. Euzenat. Ontology matching benchmarks: generation and evaluation. In *OM*, volume 814 of *CEUR Workshop Proceedings*, 2011.

[14] Y.-K. Shih and S. Parthasarathy. Scalable global alignment for multiple biological networks. *BMC Bioinformatics*, 13(S-3), 2012.

[15] P. Shvaiko and J. Euzenat. Ontology matching: State of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering*, 25(1), 2013.

[16] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel. A practical attack to de-anonymize social network users. In *IEEE Symposium on Security and Privacy*, 2010.