# Provably Fast Inference of Latent Features from Networks

## with Applications to Learning Social Circles and Multilabel Classification

Charalampos E. Tsourakakis
Harvard School of Engineering and Applied Sciences
babis@seas.harvard.edu

## ABSTRACT

A well known phenomenon in social networks is homophily, the tendency of agents to connect with similar agents. A derivative of this phenomenon is the emergence of communities. Another phenomenon observed in numerous networks is the existence of certain agents that belong simultaneously to multiple communities. An understanding of these phenomena constitutes a central research topic of network science.

In this work we focus on a fundamental theoretical question related to the above phenomena with various applications: given an undirected graph $G$, can we infer efficiently the latent vertex features which explain the observed network structure under the assumption of a generative model that exhibits homophily? We propose a probabilistic generative model with the property that the probability of an edge among two vertices is a non-decreasing function of the common features they possess. This property is true for many real-world networks and surprisingly is ignored by many popular overlapping community detection methods as it was shown recently by the empirical work of Yang and Leskovec [44]. Our main theoretical contribution is the first provably rapidly mixing Markov chain for inferring latent features. On the experimental side, we verify the efficiency of our method in terms of run times, where we observe that it significantly outperforms state-of-the-art methods. Our method is more than 2 400 times faster than a state-of-the-art machine learning method [37] and typically provides non-trivial speedups compared to BigClam [43]. Furthermore, we verify on real-data with ground-truth available that our method learns efficiently high quality labelings. We use our method to learn social circles from Twitter ego-networks and perform multilabel classification.

## Categories and Subject Descriptors

G.2.2 [**Graph Theory**]: Graph Algorithms; I.2 [**Artificial Intelligence**]: Learning

## General Terms

Theory, Experimentation

## Keywords

Machine learning; Markov chains; Graph Algorithms; Overlapping clustering; Social circles

## 1. INTRODUCTION

The major scientific concept of *pattern recognition* relies on finding the appropriate hidden variables in terms of which the patterns of a dataset are clearly described [34]. In the case of graph datasets, this major problem appears in various guises, e.g., [11, 13, 23, 32, 38] and has been attacked both by the algorithmic graph theory and the machine learning communities -among others- using different machinery. The former has focused on the ubiquitous graph clustering problem whereas the latter on learning latent classes and features. The difference between latent class models and latent feature models is that the former assume that each entity belongs exactly to a single cluster, whereas the latter allow for each entity to participate in multiple clusters. An analogous distinction also exists in the algorithmic graph theory community where the vertex set can be partitioned to either non-overlapping or overlapping sets [24].

Consider Figure 1. Each agent is represented as a vertex $v_i$, $i = 1, \ldots, 5$ and has three binary features associated with it. Each feature indicates whether the agent is interested in the news categories *business*, *technology* and *entertainment* respectively. The agents decide how to connect based on whether they share common interests. Specifically, Figure 1 shows the graph which emerges when each vertex connects to another vertex if and only if they share at least one common interest, namely a coordinate that is equal to 1 for both vertices. For instance, the single common feature between $v_1$ and $v_2$ is the *business* news. We refer to such coordinates as *common features* between the two vertices. It is clear from our toy example that given the vertex labelings and a rule on how vertices decide their connections, finding the resulting graph is easy. An interesting problem is to understand the other direction. Specifically, the question we focus on is the following:

*Under the assumption that each agent has k latent binary features, and two agents form a connection with higher probability if they share more features, how can we learn efficiently the latent features?*

Additionally to its theoretical importance, answers to this question have numerous applications. Applications of inter-

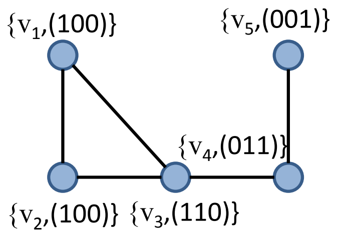

**Figure 1: A graph $G(V, E)$ on five vertices. Each vertex $v_i$ is labeled with three binary features and connects with other vertices that possess at least one common feature. For details, see text of Sections 1 and 3.**

est include learning social circles [29, 43], distributed computation [7], detecting multifunctional proteins [12], mining collaboration, word-association and protein interaction graphs [36], link prediction [32, 37] and studying cancer [22] among many others.

Our main contributions are summarized as follows.

**Modeling.** We depart from the deterministic model we described above by introducing a simple probabilistic generative model with the following property: the probability of an edge between two vertices is a non-decreasing function of their common features. This property is motivated by the recent empirical work of Yang and Leskovec [44].

**Hardness and Guarantees.** We show that maximizing the log-likelihood function is NP-hard by proving that it subsumes as a special instance the overlapping correlation clustering problem [14]. We provide combinatorial insights by connecting the machine learning problem with specific graph sub-structures. Then, we provide a rapidly mixing Markov chain for learning latent features. To the best of our knowledge, this is the first rapidly mixing chain for learning latent features from network data. Furthermore, our method sheds light on the performance of the local heuristic due to Bonchi et al. [14].

**Scalability.** Our method is more than 2 400 faster than a state-of-the-art machine learning method due to Palla et al. [37] on small sized graphs. In larger graphs the speedup becomes even more pronounced. Also, our method is trivially parallelizable.

**Accuracy and robustness to noise.** Our method is able to learn high quality latent labelings, typically achieving more than 90% precision and recall. Furthermore, our method is able to recover latent labelings from noisy similarity graphs in contrast to BigClam [43].

**Applications.** Our proposed method is able to learn social circles from Twitter ego-networks successfully. It performs comparably well or outperforms BigClam [43], the best among various competitors we compared our method against. We also use our method to perform multilabel classification from nearest-neighbor and similarity graphs.

## 2. THEORETICAL PRELIMINARIES

In this Section we present the necessary theoretical background for our main theoretical result in Section 3. The interested reader may read the book of Levin, Peres and Wilmer [30] for a detailed exposition of the concepts that follow.

| | $X_{uv} = 0$ | $X_{uv} = 1$ |
|---|---|---|
| $A_{uv} = 0$ | $1 - q$ | $1 - p$ |
| $A_{uv} = 1$ | $q$ | $p$ |

**Table 1: Conditional probabilities $\mathbf{Pr}[A_{uv}|X_{uv}]$. Indicator variable $X_{uv}$ takes the value 1 if and only if vertices $u, v$ share at least 1 out of $k$ features, i.e., $\omega^T(u)\omega(v) \geq 1$. For details, see text of Section 3.**

DEFINITION 1 (COUPLING). *A coupling of two probability distributions $\mu, \nu$ is a pair of random variables $X, Y$ defined on a single probability space such that the marginal distribution of $X$ is $\mu$ and the marginal distribution of $Y$ is $\nu$.*

The mixing time measures the time required by a Markov chain to approach the stationary distribution. A precise definition follows. We remind the reader that the total variation distance between two probability distributions $\mu, \nu$ is $|\mu - \nu|_{TV} = \frac{1}{2}\sum_{x\in\Omega} |\mu(x) - \nu(x)|$.

DEFINITION 2 (MIXING TIME). *The mixing time is defined as*

$$t_{mix}(\epsilon) = \min\{t : \max_{x\in\Omega} |P^t(x, \cdot) - \pi|_{TV} \leq \epsilon\}$$

*where $\pi$ is the stationary distribution and $P^t(x, \cdot)$ is the probability distribution of the state of the chain starting at state $x$ after $t$ steps.*

We invoke the following result due to Bubley and Dyer in Theorem 2.

THEOREM 1 (PATH COUPLING [15]). *Suppose the state space $\Omega$ of a Markov chain is the vertex of a graph with length function $l$ defined on edges. Let $\rho$ be the corresponding path metric, namely*

$$\rho(x, y) = \min\{\textit{length of P:P a path from x to y}\}.$$

*Suppose that for each edge $\{x, y\}$ there exists a coupling $(X_1, Y_1)$ of the distributions $P(x, \cdot)$ and $P(y, \cdot)$ such that*

$$\mathbb{E}[\rho(X_1, Y_1)|X_0 = x, Y_0 = y] \leq \rho(x, y)e^{-\alpha}.$$

*Then,* $t_{mix}(\epsilon) \leq \left\lceil \frac{-\log\epsilon+\log\left(diam(\Omega)\right)}{\alpha} \right\rceil$.

## 3. PROPOSED METHOD

**Generative model.** Consider again the example in Figure 1 and suppose for instance that the number of latent features is not a priori known. A simple argument shows that the smallest number of latent features $k^*$ that can perfectly explain the observed graph is three. To see why, assume for the sake of contradiction $k^* = 2$. Since there are no isolated vertices, there exist three possible labelings $\{(01), (10), (11)\}$. Notice that vertex $v_5$ cannot be labeled as (11) since it is not connected to every other vertex. Hence, without loss of generality it is labeled (01). Since $v_1, v_2, v_3$ are not connected with $v_5$, they have to be labeled by (10). But then since $v_1, v_3$ have the same label, they should see exactly the same (non)-neighborhood which contradicts the observed graph structure. As it is obvious from Figure 1(a), three labels suffice to explain perfectly the graph structure.

We depart from this deterministic generative model as follows. Let $\omega : V \to \{0,1\}^k$ be the latent vertex labeling. Define for each vertex $v$ the set $\mathcal{C}(v)$ of non-zero coordinates. This set intuitively corresponds to the *interests* of $v$. Vertex $v$ establishes its connections with other vertices based on the following simple rule: for each vertex $u$ such that $|\mathcal{C}(v) \cap \mathcal{C}(u)| \geq r$ it adds an edge $(u,v)$ with probability $p$. If $|\mathcal{C}(v) \cap \mathcal{C}(u)| \leq r-1$ it adds an edge $(u,v)$ with probability $q < p$. Notice that under this model, a vertex $v$ with no interests ($\mathcal{C}(v) = \emptyset$), connects to each $u \in V \backslash \{v\}$ independently with probability $q$. In the rest of the paper, we set $r = 1$, which results already in a simple yet non-trivial probabilistic model. We define for each pair of vertices $u, v$ the indicator variable $X_{uv} = 1(\omega^T(u)\omega(v) \geq 1) = 1 - 1(\omega^T(u)\omega(v) = 0)$. Table 1 shows for any pair of vertices $u, v$ the probability of the edge $(u,v)$ conditioned on the value $X_{uv}$. Specifically, $\mathbf{Pr}[A_{uv} = 1 | X_{uv} = 1] = p \in (0,1)$ and $\mathbf{Pr}[A_{uv} = 1 | X_{uv} = 0] = q \in (0,1)$.

**Maximum likelihood function.** The likelihood function of a graph $G(V,E)$ given the latent labeling is

$$\mathbf{Pr}[G|\omega] = \prod_{(u,v)\in E(G)} p^{X_{uv}}(1-p)^{1-X_{uv}} \times \prod_{(u,v)\notin E(G)} q^{X_{uv}}(1-q)^{1-X_{uv}} \Leftrightarrow$$

$$\log_2\big(\mathbf{Pr}[G|\omega]\big) = C(n,m,p,q) + \sum_{(u,v)\in E(G)} X_{uv}\log_2\big(\frac{p}{1-p}\big) + \sum_{(u,v)\notin E(G)} (1-X_{uv})\log_2\big(\frac{1-q}{q}\big)$$

where $C(n,m,p,q) = m\log_2(1-p) + \big(\binom{n}{2} - m\big)\log_2 q$ is a constant that does not depend on the labeling $\omega$. We show that maximizing $\mathbf{Pr}[G|\omega]$ over $\omega \in \Omega = 2^{[nk]}$, the set of all possible binary labelings of $n$ vertices with $k$ binary features, is in general NP-hard.

**Objective** Let's consider the special case $p = 1 - q = \frac{2}{3}$. Then, maximizing the likelihood function becomes equivalent to

$$\max_{\omega:V\to\{0,1\}^k} \sum_{(u,v)\in E(G)} 1(\omega^T(u)\omega(v) \geq 1) + \sum_{(u,v)\notin E(G)} 1(\omega^T(u)\omega(v) = 0) \qquad (1)$$

We refer to the function we aim to maximize as $h : \Omega \to \mathbb{R}$. This objective was introduced by Bonchi et al. [14] and is known as the overlapping correlation clustering (OCC) problem. Our analysis provides a probabilistic interpretation of OCC [14]. Furthermore, we directly obtain that maximizing the likelihood function for a given graph under our generative model is NP-hard. This is stated as the next lemma.

Lemma 1 ([14]). *Optimizing 1 in general is NP-hard.*

In the following we study certain basic combinatorial aspects of Objective 1. First, notice that trivially the optimum of Objective 1 is at most $\binom{n}{2}$. We refer to this value as *perfect* score. Our first observation is that when the number of features $k$ is at least the number of edges, we can achieve a perfect score. Specifically when $k \geq m$, for each edge $e = (u,v) \in E$ we create a binary feature $f_e$. We set the coordinate $f_e$ to 1 only for the endpoints $u, v$. For the rest of the vertices, $f_e$ is set to 0.

Consider the star graph $K_{1,n-1}$ on $n$ vertices, an instructive special case. How many features are there needed to obtain a perfect score? Clearly, if we use $k = n-1$ features, then there is a labeling achieving perfect score. The center vertex is labeled as $\underbrace{1\ldots1}_{n-1 \text{ ones}}$ and for all $i = 1, \ldots, n-1$ the $i$-th leaf is labeled as $e_i$, where $e_i$ is a vector whose coordinates are zero except for the $i$-th coordinate which is equal to 1. Can we use $k < n-1$ features and still have a perfect score? The answer is negative. To see why, assume that there exists $k < n-1$ such that we can label the vertices with $k$ features and obtain a perfect score. We can assume without any loss of generality (otherwise the score is not perfect) that all the edges of the star tree are satisfied, namely each leaf shares at least one feature with the center vertex. Consider the labels of the $n-1$ leaves and specifically their non-zero coordinates that agree with the center vertex. By the pigeonhole principle, since $k < n-1$ at least one pair of leaves will have to share a common feature. But since these two leaves are not connected by an edge, at least one non-edge is not satisfied and hence the score cannot be perfect.

We wish to outline two facts. We observe that the latent labeling sheds light on the overlapping community structure of the graph. In the case of the star, we have $n-1$ communities overlapping on the center vertex, which is perfectly reflected by the labeling achieving a perfect score. Secondly, the argument above can be generalized to provide a lower bound on the number of features needed to obtain a perfect score. This is stated as the next lemma.

Lemma 2. *Let $K_{\alpha,\beta}$ be an induced bipartite clique in $G(V,E)$. Then, at least $\alpha\beta$ features are required to achieve a perfect score.*

The proof is omitted as it is similar to the argument for the star graph $K_{1,n-1}$. The above lemma implies that if $K_{\alpha,\beta}$ is an induced bipartite clique that maximizes the product $\alpha\beta$ over all possible induced bipartite cliques, $\alpha\beta$ is a lower bound on the bound of features we need to explain the graph. On the other hand, as we noticed above $m$ is an upper bound. Hence the bounds are tight, as we can see from the star graph $K_{1,n-1}$. Additionally, a simple corollary of Lemma 2 is that the number of features needed is at least the diameter of the graph, a computationally tractable graph parameter.

Corollary 1. *At least $D$ features are required to achieve a perfect score, where $D$ is the diameter of the graph.*

Finally, we observe that there exists a trivial constant factor approximation algorithm.

Lemma 3. *For any $k \geq 1$ there exists a trivial constant approximation factor.*

Proof. If $m = \Theta(n^2)$, then let $f(v) = \underbrace{1\ldots1}_{k \text{ ones}}$ for all $v \in V(G)$, otherwise let $f(v) = \underbrace{0\ldots0}_{k \text{ zeros}}$. $\square$

Obtaining a polynomial time approximation scheme (PTAS) is an interesting direction. Here we design a rapidly mixing chain.

**Algorithm 1** Metropolis Chain

```
Input: Graph $G$, mixing parameter $c$, number of steps $T$
1: Let $\omega_0 : V \to \{0,1\}^k$ be an arbitrary initial labeling.
2: for $t \leftarrow 1$ to $T$ do
3:     Choose a vertex $u$ uniformly at random from $V(G)$ and a $k$-labeling $x$ uniformly at random from $2^{[k]}$
4:     Perform the transition from $\omega_{t-1}$ to $\omega_t$ such that $\omega_t(u) = x$ and $\omega_t(v) = \omega_{t-1}(v)\ \forall v \neq u$ with probability $\min\left(1, e^{c\left(h(\omega_t)-h(\omega_{t-1})\right)}\right)$.
5: end for
6: Output the best seen state.
```

**Proposed method.** Our proposed algorithm is shown in Algorithm 1. Our algorithm maximizes the function $h : \Omega \to \mathbb{R}$, see Equation (1), using a Metropolis chain, which takes as input the graph $G$, the number of steps $T$, and a parameter $c \geq 0$ which controls the trade-off between the search space efficiency and the quality of the optimization. In the following we interchangeably refer to a binary labeling as a state. For a state $\omega$ we denote with $\omega(v)$ the labeling of vertex $v$. The metric we choose is the Hamming metric. Specifically, the distance between two states $\omega_1 \neq \omega_2$ is the number of vertices in which the two states differ, i.e., $dist(\omega_1, \omega_2) = |\{u \in V : \omega_1(u) \neq \omega_2(u)\}|$. Notice that the diameter of the state space is exactly $n$, i.e., $diam(\Omega) = n$.

Even if Metropolis chains are treated in standard textbooks [30], it is worth explaining parameter $c$ in further detail as it highlights the significance of our theoretical result. In the limit $c \to +\infty$ the chain resembles a deterministic hill climb algorithm and hence is limited by local optima which can result in a poor latent labeling. This is essentially the method of Bonchi et al. [14], even if it is not explicitly stated in their paper. This also explains the main drawback of their method, namely being prone to local optima. At the other extreme, as $c \to 0$ the search over the state space becomes efficient but in vain: the latent labelings we aim to find are not favored by the stationary distribution since the latter becomes uniform. Our main theoretical result is that there exists a non-trivial choice of $c$ for which the Metropolis chain both *mixes rapidly* even if the state space is exponentially large and *favors the states that optimize our objective*. This is stated as Theorem 2. In the following, we assume that $k = \Theta(1)$ which results in a practical $\tilde{O}(n)$ algorithm.

THEOREM 2. *There exists a non-trivial range for the parameter $c = c(n)$ for which the Metropolis chain mixes rapidly, i.e., $t_{mix}(\epsilon) = O\left(n \log_2\left(\frac{n}{\epsilon}\right)\right)$ for any $\epsilon > 0$.*

PROOF. Let $X_t, X'_t$ be two copies of the chain starting from states $\omega_1, \omega_2 \in \Omega$, such that $dist(\omega_1, \omega_2) = 1$. Let $u \in V$ be the single vertex for which $\omega_1(u) \neq \omega_2(u)$. We consider the following coupling:

- Chain $X_t$ chooses a vertex $v$ and a string $x \in \{0,1\}^k$, both uniformly at random.
- Chain $X'_t$ chooses exactly the same vertex-string pair $(v, x)$.

Let $\omega, \omega'$ be the two new candidate states for the first $(X_t)$ and the second chain $(X'_t)$ respectively. Now we set the joint transition probability distribution in such a way that $X_t, X'_t$ stay as close as possible while respecting the marginals.

Case I: $\mathbf{Pr}[\omega_1, \omega] \geq \mathbf{Pr}[\omega_2, \omega']$: If the first chain $(X_t)$ performs the switch, then the second chain $(X'_t)$ performs also a switch with probability $\frac{\mathbf{Pr}[\omega_2,\omega']}{\mathbf{Pr}[\omega_1,\omega]}$. If the first chain does not perform a switch, so does not the second chain too. Notice that the marginal for the second chain is correct. To see why, the probability that the second chain $X'_t$ performs a switch is

$$\mathbf{Pr}[\omega_1, \omega]\frac{\mathbf{Pr}[\omega_2, \omega']}{\mathbf{Pr}[\omega_1, \omega]} + (1 - \mathbf{Pr}[\omega_1, \omega])0 = \mathbf{Pr}[\omega_2, \omega'] .$$

Case II: $\mathbf{Pr}[\omega_1, \omega] < \mathbf{Pr}[\omega_2, \omega']$: If the first chain $(X_t)$ performs the switch, so does the second chain $(X'_t)$. If the first chain does not perform a switch, the second chain performs a switch with probability $\frac{\mathbf{Pr}[\omega_2,\omega'] - \mathbf{Pr}[\omega_1,\omega]}{1-\mathbf{Pr}[\omega_1,\omega]}$. Again, notice that the marginal for the second chain is correct. The probability that the second chain $X'_t$ performs a switch is indeed what it should be:

$$\begin{aligned}\mathbf{Pr}[\omega_1, \omega] \times 1 + (1 - \mathbf{Pr}[\omega_1, \omega])\frac{\mathbf{Pr}[\omega_2, \omega'] - \mathbf{Pr}[\omega_1, \omega]}{1 - \mathbf{Pr}[\omega_1, \omega]}& \\ = \mathbf{Pr}[\omega_2, \omega'] .&\end{aligned}$$

Now we study the expected distance between the two chains after one step. Notice that the distance may remain the same, but it also may increase by 1, or decrease by 1. Let $\mathcal{I}, \mathcal{D}$ be the events for the two latter cases. Then, the expected distance after a single step of the chain is

$$\mathbb{E}[d_{t+1}|d_t = 1] = 1 + 2\mathbf{Pr}[\mathcal{I}] - \mathbf{Pr}[\mathcal{D}] .$$

Since we wish to upper bound $\mathbb{E}[d_{t+1}|d_t = 1]$, we appropriately bound $\mathbf{Pr}[\mathcal{D}]$ and $\mathbf{Pr}[\mathcal{I}]$. Before that, we make a simple observation that we use for both bounds. Consider any labeling $\omega : V \to \{0,1\}^k$ and suppose we change the labeling of a single vertex $u$, resulting in the new labeling $\omega'$. Then, clearly $|h(\omega) - h(\omega')| \leq n - 1$ and the upper bound is tight.

Lower-bound $\mathbf{Pr}[\mathcal{D}]$ : If the chosen vertex $v$ is vertex $u$, then the distance cannot increase since $\omega = \omega'$. Specifically, the distance either remains 1 or decreases to 0. The latter happens when both chains perform the switch. Let this event be $\mathcal{S}$ Given our coupling the probability that both perform the switch is $\mathbf{Pr}[\mathcal{D}] = \min\left(\mathbf{Pr}[\omega_1, \omega], \mathbf{Pr}[\omega_2, \omega]\right)$. To see why, notice that if $\mathbf{Pr}[\omega_1, \omega] \geq \mathbf{Pr}[\omega_2, \omega]$ then the probability that both chains switch is due to our coupling $\mathbf{Pr}[\omega_1, \omega]\frac{\mathbf{Pr}[\omega_2,\omega]}{\mathbf{Pr}[\omega_1,\omega]} = \mathbf{Pr}[\omega_2, \omega]$. Similarly, if $\mathbf{Pr}[\omega_1, \omega] < \mathbf{Pr}[\omega_2, \omega]$ then the probability is $\mathbf{Pr}[\omega_1, \omega] \times 1$. To lower bound $\mathbf{Pr}[\mathcal{D}]$ we use the inequality $e^{-x} \geq 1 - x$ which is true for $x \geq 0$. Since the probability of choosing the vertex $u$ on which the two states $\omega_1, \omega_2$ disagree is $\frac{1}{n}$, we obtain that $\mathbf{Pr}[\mathcal{S}] \geq \frac{1}{n}e^{-cn} \geq \frac{1}{n} - c$.

Upper bound $\mathbf{Pr}[\mathcal{I}]$: Suppose that the chosen vertex $v$ is not $u$, i.e., i.e., the two chains agree on the labeling of $v$, $\omega_1(v) = \omega_2(v)$. Then, if both chains perform the switch the distance remains 1, otherwise if exactly only one of the two chains performs the switch the distance increases to 2. First we notice that if $\mathbf{Pr}[\omega_1, \omega] \geq \mathbf{Pr}[\omega_2, \omega']$ the probability that only the first chain performs the switch is $\mathbf{Pr}[\omega_1, \omega]\left(1 - \frac{\mathbf{Pr}[\omega_2,\omega']}{\mathbf{Pr}[\omega_1,\omega]}\right) = \mathbf{Pr}[\omega_1, \omega] - \mathbf{Pr}[\omega_2, \omega']$. If $\mathbf{Pr}[\omega_1, \omega] < \mathbf{Pr}[\omega_2, \omega']$ then the probability that only the second chain performs the switch is $\frac{\mathbf{Pr}[\omega_2,\omega'] - \mathbf{Pr}[\omega_1,\omega]}{1-\mathbf{Pr}[\omega_1,\omega]}\Big(1 - \mathbf{Pr}[\omega_1, \omega]\Big) = \mathbf{Pr}[\omega_2, \omega'] - \mathbf{Pr}[\omega_1, \omega]$. We can write this in a compact way by saying that the probability that exactly one of the two chains performs the switch is

$$F = |\mathbf{Pr}[\omega_1, \omega] - \mathbf{Pr}[\omega_2, \omega']\,|.$$

We upper-bound $F$ by considering one of the two probabilities as large as possible, namely 1, and the second as small as possible, namely $e^{-cn}$. Therefore,

$$F \leq 1 - e^{-cn} \leq cn.$$

Notice that the probability that we choose $v \neq u$ is $\frac{n-1}{n}$ and the probability we choose a labeling $x \in \{0,1\}^k$ such that $x \neq \omega_1(v) = \omega_2(v)$ is $\frac{2^k-1}{2^k}$. Therefore, we obtain

$$\mathbf{Pr}[\mathcal{I}] = \frac{n-1}{n}\frac{2^k-1}{2^k}F < F \leq cn.$$

Therefore, by choosing $c = \frac{1}{3n^2}$, for any $n \geq 4$ we obtain the following valid inequality

$$\begin{aligned}\mathbb{E}[d_{t+1}|d_t = 1] &= 1 + 2\mathbf{Pr}[\mathcal{I}] - \mathbf{Pr}[\mathcal{D}] \\ &\leq 1 + 2cn - (\frac{1}{n} - c) \leq 1 - \frac{1}{4n} = \beta.\end{aligned}$$

Therefore, by Theorem 1 we obtain that the mixing time

$$t_{\text{mix}}(\epsilon) \leq \frac{\log_2\left(\frac{n}{\epsilon}\right)}{1-\beta} = O\Big(n\log_2\left(\frac{n}{\epsilon}\right)\Big).$$

Finally, the states that maximize Objective 1 are favored by this choice of $c$. By Lemma 3, the optimal states achieve an objective of $Dn^2$ for some constant $D$. Hence, the Metropolis chain for $c = \frac{1}{3n^2}$ assigns to them value $e^{D/3}$. $\square$

We remark that our method runs in polynomial time for $k \leq b\log_2 n$ where $b > 0$ is any constant. Specifically, for $k = b\log_2 n$ we obtain a $\tilde{O}(n^{b+1})$ algorithm due to the implicit $O(2^k)$ term in the search of labelings for any vertex.

## 4. EXPERIMENTAL RESULTS

### 4.1 Experimental Setup

**Implementation and competitors.** Our algorithm is implemented in MATLAB. We remove the random bits needed to select a vertex in each round uniformly at random by selecting an arbitrary ordering of vertices and considering changes in their labelings according to that order. Empirically this appears to have little effect -if any- on the results. Furthermore it avoids the coupon collector problem according to which $O(n\log n)$ draws of vertices are needed to consider all vertices at least once with high probability. A detailed understanding of this systematic scan is left as an interesting problem [33]. Finally, we set the maximum number of steps equal to $\lceil n\log n\rceil$. We terminate the chain either if it reaches the maximum number of steps or if for $n$ steps it stays on the same state. We refer to our method as *FINLAnd* (Fast INference of LAtent features). We compare our method against methods that are publicly available: ILA [37] (MATLAB), BigClam [43] (C++) and CFinder [3] which is provided as a (.EXE) file. We received an improved MATLAB implementation of ILA than the one that is on the Web (as of 08/2014) upon contacting the authors of [37]. All methods were executed on a single machine with Intel Xeon CPU at 2.83GHz and 50GB RAM. As we describe below we used datasets with available ground-truth. In all executions the true number of latent features was passed as an argument to all methods. While there exist several heuristics to approximate the number of features $k$, we consider this problem worth a separate project.

**Datasets.** We used publicly available datasets in our experiments. Specifically, we use ego-networks from Twitter[1] [29], EMOTIONS and FLAGS, two Mulan[2] Multi-Label Learning datasets, and the MNIST digit database [1]. Specifically, EMOTIONS and FLAGS have $(593, 6)$ and $(194, 7)$ (points, features) respectively. We create a similarity graph out of each dataset by adding an edge between two points that share at least one common feature. From MNIST we extract a perfectly balanced dataset of 10 000 digits (1 000 digits for each digit 0,1,. . . ,9). Each digit is a 28×28 matrix which is converted in a 1-dimensional vector with 784 coordinates. We create $k'$-nearest-neighbor graphs from this cloud of points for radically different $k'$ values which result in graphs with $O(n^2)$ and $O(n)$ edges. Finally, all graphs were made *undirected* by replacing each directed edge within an undirected one and *simple* by removing multiple edges.

**Evaluation.** We use the well-known measures of precision ($P$) and recall ($R$) to evaluate the performance of the algorithms. Specifically, for any labeling $\omega : V \to \{0,1\}^k$ we define the set $\mathcal{F}(\omega) = \{(u,v) : \omega^T(u)\omega(v) \geq 1\}$. Also, let $g$ be the ground-truth labeling. Then,

$$\mathrm{P}_\omega = \frac{|\mathcal{F}(\omega) \cap \mathcal{F}(g)|}{|\mathcal{F}(\omega)|} \qquad \text{and} \qquad \mathrm{R}_\omega = \frac{|\mathcal{F}(\omega) \cap \mathcal{F}(g)|}{|\mathcal{F}(g)|}.$$

We omit the indices when it is clear to which labeling $\omega$ we refer to.

### 4.2 Experimental findings

Before we present our findings in detail, we summarize them.

**Mixing parameter $c$.** By experimentation, we found that for the given number of steps that we run the chain, constant values of $c$, e.g., $c = 1/2$, achieve better performance than the value predicted by Theorem 2. This suggests that injecting a small amount of randomness in the deterministic hill climbing algorithm works well. This discrepancy between theory and practice indicates that either one has to increase the number of steps for which the chain runs or adopt a simulated annealing approach [25]. We experimented with the former approach and indeed we found that the output quality increases when $c = \frac{1}{3n^2}$ as we let the chain run for $b\lceil n\log n\rceil$ steps where $b > 0$ is a constant.

[1]http://snap.stanford.edu/data/egonets-Twitter.html
[2]mulan.sourceforge.net/datasets.html

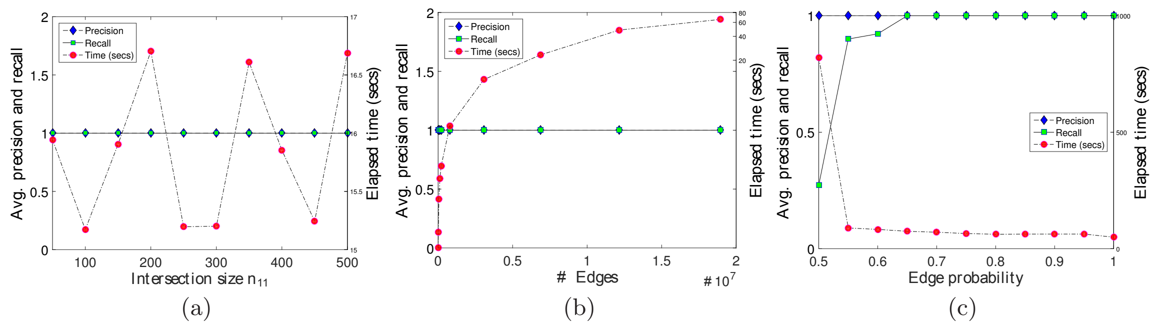

**Figure 2: Average precision and recall (left $y$-axis) and average running time (right $y$-axis) versus (a) the intersection size $n_{11}$, (b) the number of edges and (c) the edge probability. Each point in the plots is the average over 25 experiments (5 randomly generated graphs $\times$ 5 random initial labelings). For details on each experiment, see text of Section 4.3**

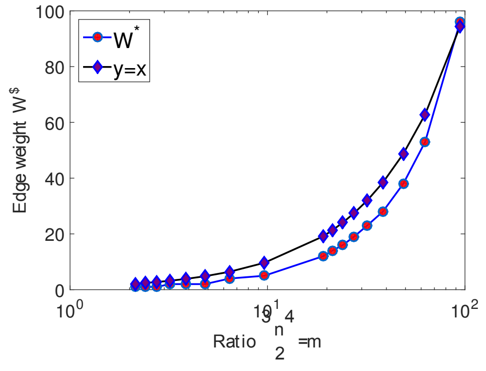

**Figure 3: Figure plots in semilog the weight $W^*$ for which we obtain at least 90% precision and recall at the same time versus the ratio $\binom{n}{2}/m$. Included is the plot of the identity function $W = \binom{n}{2}/m$ annotated properly in the legend. For details, see text of Section 4.3.**

**Efficiency.** Our proposed method is able to learn latent features efficiently. For graphs with few hundreds of edges, our method is more than 2 400 times faster than the state-of-the-art ILA method [37]. The speedup becomes even more pronounced in large networks since ILA does not scale at all.

**Robustness.** Our proposed method is both robust to the initial labeling and to noise. Specifically, it is able to optimize the objective independently on how the vertices are initially labeled and furthermore even when a constant fraction of the edge set changes ($\sim$ 15%) our algorithm is able to recover the latent features with high accuracy. This is not always true for other competitors.

**Best competitor.** Our algorithm either outperforms or performs comparably well with BigClam [43], the best among the various methods we tried for the task of overlapping community detection. Surprisingly, even if our method is coded in MATLAB and BigClam in C++, it runs typically faster than BigClam.

**Sparse graphs and a rule-of-thumb.** Optimizing Objective 1 works well for dense graphs. To see why, consider the case of sparse graphs, namely $m = \tilde{O}(n)$. This is the typical case for most real-world networks. Since $m = o(n^2)$, the first term of Objective 1 is asymptotically vanishing, essentially contributing nothing to the optimal value. This means that even if we could optimize Objective 1 exactly, the optimal solution would not provide any insights. For this purpose, we propose for the case of sparse graphs a variation of Objective 1. Specifically, we introduce a positive real parameter $W$.

$$\max_{\omega:V\to\{0,1\}^k} \sum_{(u,v)\in E(G)} W1(\omega^T(u)\omega(v) \geq 1) + \sum_{(u,v)\notin E(G)} 1(\omega^T(u)\omega(v) = 0) \tag{2}$$

We provide a surprisingly good rule-of-thumb on how to choose $W$. This value is equal to $\binom{n}{2}/m$. This value for $W$ results in a normalized version of Objective 1.

**Social circles.** Using our rule-of-thumb we apply our Metropolis chain adapted for Objective 2 on the problem of clustering ego-networks. Notice that the only thing that changes in the pseudocode is function $h()$, which now becomes the function in Objective 2. We obtain *high* quality results for this graph mining problem, typically outperforming BigClam [43].

**Detailed remarks about competitors.** Among the numerous works for learning latent features that appear within the machine learning community, e.g., [16, 17, 20, 37], we choose the ILA method [37] as our competitor, since it was published fairly recently. We were able to run ILA only on graphs with few hundreds of edges. To be precise, the largest graph on which ILA produced results within a reasonable amount of time was a graph with 40 vertices and few hundreds of edges. Our algorithm required 0.02 seconds whereas ILA 49.1 seconds. It is worth outlining that not only our method is 2 455 times faster than ILA but also the output quality of our algorithm is better. For the record, ILA produced a good labeling achieving precision and recall equal to 0.99 and 0.93 respectively. For larger graphs, ILA fails completely to produce a labeling within a reasonable amount of time. For instance, ILA has not completed its execution on the EMOTION graph which has $\sim$ 600 nodes after several days. The intermediate results are of poor quality, achieving less than 60% precision and recall.

For the task of finding overlapping communities, we experimented with CFinder [3], BigClam [43], a method based on nonnegative matrix factorization. The BigClam method is the best performing method. CFinder is unable to de-

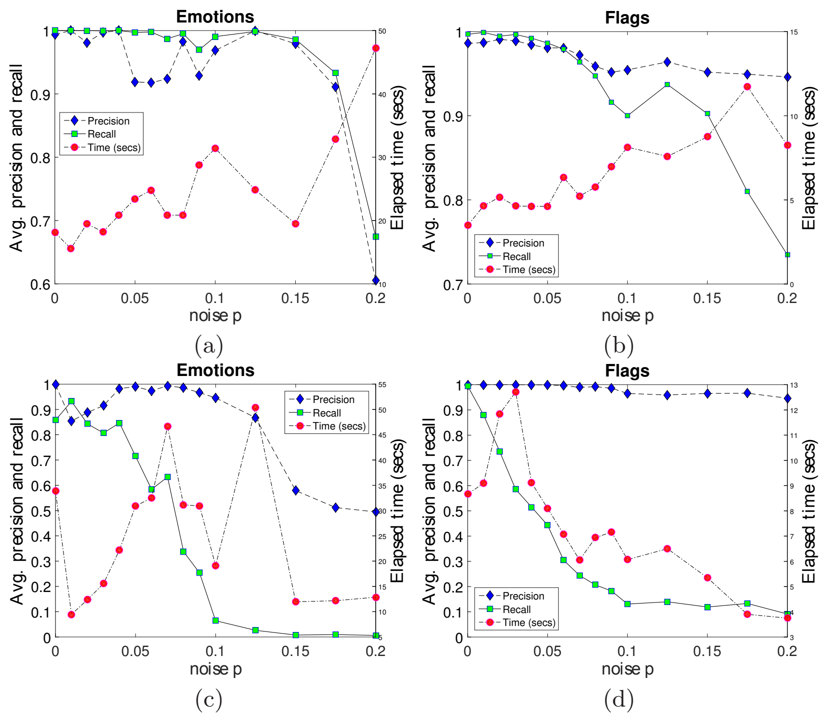

**Figure 4: Average precision and recall (left $y$-axis) and average running time (right $y$-axis) versus sparsification parameter $p$ on the Flags and Emotions similarity graphs. (a)(b) FINLAnd, (c)(d) BigClam. For details, see text of Section 4.4.**

tect overlapping communities when the overlaps are dense, verifying the claim of Yang and Leskovec [44]. More importantly, CFinder does not scale to large graphs. Specifically, CFinder runs for hours on graphs with several thousands of edges, when other competitors are able to output within seconds. This is expected to happen when there exist strongly overlapping dense subgraphs, which as Yang and Leskovec point out is the typical case [44].

In the following sections which contain the results on real-world data, we include full results for [43], as it was the only competitor that could successfully complete for all experiments within a reasonable amount of time.

## 4.3 Tiles

Yang and Leskovec liken tiles and communities: characterizing precisely for each vertex to which communities it belongs to resembles the setting where some number of tiles are placed on the top of each other in an overlapping manner and the goal is to move them without breaking any of them [44].

**Setting.** We use two overlapping tiles or equivalently the 2-dimensional hypercube as our latent space. Specifically, we assume there exist two sets of vertices $S_1, S_2$ such that $|S_1| = n_{01}+n_{11}, |S_1| = n_{10}+n_{11}$ and $|S_\cap| = |S_1 \cap S_2| = n_{11}$. For each pair of distinct vertices $u, v$ such that $u, v \in S_\cap$ we add an edge with probability $p_{11}$. For every other pair of vertices in $S_1$ ($S_2$) we add an edge with probability $p_{01}$ ($p_{10}$). The latent labeling for vertices in $R_1 = S_1 \backslash S_\cap$, $S_\cap$, $R_2 = S_2 \backslash S_\cap$ is $(01), (11), (10)$ respectively. Notice that this labeling is not unique, the labels of $R_1, R_2$ can be exchanged. We explore various aspects of this setting.

**Parameter $n_{11}$ and scalability.** We generate dense graphs with $O(n^2)$ edges by fixing $p_{01} = p_{10} = 0.75 < p_{11} = 0.95$ in order to simulate the "denser-overlaps" phenomenon [44]. The results are shown in Figures 2 (a),(b). Later, we show how to deal with sparse graphs successfully.

Specifically, Figure 2(a) ranges the intersection size $n_{11} \in \{1, 100 : 50 : 500\}$ where $a : b : c$ is the standard MATLAB notation for indicating the starting value $a$, the step $b$ and the final value $c$. The total number of vertices is fixed to 5 000. For each value $n_{11}$ we run our method *five* times on *five* randomly generated graphs. We compute for each of the 25 experiments the precision and recall and we plot the average precision and recall (left y-axis) versus $n_{11}$. Results are perfectly concentrated: average precision and recall (left y-axis) are always 1 for all $n_{11}$ values. Furthermore, there does not appear to be any correlation between $n_{11}$ and the average running time (right y-axis): running time fluctuations at the order of one second are most likely due to randomness.

Figure 2(b) shows the results we obtain as we increase the number of vertices in the graph, for a fixed intersection size $n_{11} = 50$. We range $n \in \{100, 250 : 250 : 1000, 2000 : 2000 : 10000\}$. On the left $y$-axis we plot again the average precision and recall versus the averaged number of edges over the five random graphs we generate for each value $n$. On the right $y$-axis the average running time. To emphasize the scalability we plot versus the number of edges ($x$-axis). Our method is able to recover the groundtruth labeling in all 25 experiments for all $n$ values. Furthermore we observe that it scales well as the size of the graph grows.

**Running time depends not only on size but structure as well.** The following experiment outlines an interesting aspect of our method: its running time depends not only on the graph size but also on the graph structure. Equivalently, it is strongly indicated by our experimental results that if the graph cut structure exhibits good clustering properties, then our chain can find it easily and decide to stop within a linear number of steps. We use again the setting of two tiles. We set $p_{01} = p_{10} = p_{11} = p$ and the total number of vertices to $n = 10\,000$. We observe an interesting transition at $p = 1/2$.

Figure 2(c) plots the average precision and recall (left $y$-axis) and the average running time (right $y$ axis) versus the edge probability $p$. We range $p$ in the set $\{0.5 : 0.05 : 1\}$. The intersection size is fixed for all values of $p$ to $n_{11} = 100$. Again, we perform for each value $p$, twenty-five experiments in total. When $p = 1/2$ the average running time is 820 seconds. Additionally to the slow performance for $p = 1/2$, the running time exhibits significant variance. Specifically, the standard deviation is 112 seconds. It is worth emphasizing that the datapoint corresponding to the average running time for $p = 1/2$ in Figure 2(c) is one of the two datapoints in all plots that does not exhibit strong concentration. The other one is the average recall, again for $p = 1/2$, which is equal to 0.27 with standard deviation equal to 0.06. Notice that that for all values of $p$ greater than 1/2 the running time is significantly smaller even if the number of edges grows. For instance, when $p = 1$ the average running time is 49.4 seconds with a standard deviation of 0.29 seconds. This is happening because for $p = 1/2$ the existing density within the communities is not large enough, hence the optimal value of the objective does not correspond a latent labeling capturing the community structure. We deal with this immediately in the following.

| Pair | FINLand | | | BigClam [43] | | |
|---|---|---|---|---|---|---|
| | $P$ | $R$ | $t$ | $P$ | $R$ | $t$ |
| (0,1) | 0.98 | 0.87 | 4.65 | 0.99 | 0.88 | 21.25 |
| (0,7) | 0.97 | 0.84 | 4.23 | 0.96 | 0.81 | 25.49 |
| (0,8) | 0.97 | 0.83 | 4.62 | 0.93 | 0.78 | 22.31 |
| (0,9) | 0.96 | 0.83 | 4.18 | 0.95 | 0.80 | 24.76 |
| (1,4) | 0.96 | 0.88 | 4.47 | 0.99 | 0.80 | 25.54 |
| (1,6) | 0.94 | 0.87 | 3.56 | 0.98 | 0.75 | 22.67 |
| (1,7) | 0.92 | 0.86 | 4.18 | 0.94 | 0.77 | 24.57 |
| (1,9) | 0.95 | 0.86 | 3.19 | 0.96 | 0.74 | 25.63 |
| (3,4) | 0.95 | 0.85 | 4.66 | 0.97 | 0.83 | 25.64 |
| (3,6) | 0.95 | 0.83 | 4.07 | 0.96 | 0.85 | 20.39 |
| (3,7) | 0.95 | 0.82 | 3.57 | 0.97 | 0.80 | 25.01 |
| (6,7) | 0.98 | 0.88 | 4.11 | 0.99 | 0.87 | 23.41 |
| (6,9) | 0.96 | 0.81 | 5.38 | 0.99 | 0.73 | 25.02 |

**Table 2: Table shows the performance of FINLand and Bigclam in terms of precision (P), recall (R) and running time (t) for several pairs of digits. For details, see text of Section 4.4.**

**Sparse graphs and a rule-of-thumb.** Lemma 3 shows that when the graph is sparse Objective 1 does not necessarily yield insightful solutions. As we have mentioned again, this happens since the first term corresponding to the summation over the edges is asymptotically negligible with respect to the second term in the regime when $m = o(n^2)$. For this purpose, we introduce Objective 2 which gives more importance satisfying an edge rather than a non-edge. This is achieved by the weight parameter $W \geq 1$. However, given a graph $G$ it is not a priori clear how to choose $W$ in order to learn a good latent vertex labeling. Here, we provide a simple yet well-performing rule-of-thumb on how to choose $W$. For this purpose we perform the following experiment. We generate two tiles on 1 000 vertices. We range $p_{01} = p_{10}$ in the set of values $\{0.01{:}0.01{:}0.1,.2{:}0.1{:}.9\}$. Notice we choose an increased level of granularity when $p_{01} = p_{10}$ is small. The intersection probability is $p_{11} = p_{01} + 0.1$ and the intersection size between the two equal-sized tiles is 20. For each value $p_{01}$ we search for the value $W^*_{\text{true}}$ that results in both precision and recall greater than 90%. Since every time that either the precision or the recall are less than 90%, we increase the weight $W$ by one $|W^* - W^*_{\text{true}}| \leq 1$. Figure 3 plots in semilog $W^*$ versus the inverse density ratio $\binom{n}{2}/m$ where $m$ ranges as the edge probabilities change. We observe a striking phenomenon that we verified for various other settings of the parameters as well and that we exploit later on real data. Specifically, we plot the function $W^* = \binom{n}{2}/m$ which is very close to the observed measurements. Notice that this specific value of $W^*$ results equivalently in a normalized version of our objective:

$$\max_{\omega: V \to \{0,1\}^k} \frac{1}{m} \sum_{(u,v)\in E(G)} 1(\omega^T(u)\omega(v) \geq 1) + \frac{1}{\binom{n}{2}} \sum_{(u,v)\notin E(G)} 1(\omega^T(u)\omega(v) = 0)$$

## 4.4 Classification

**Digit classification.** We choose several pairs of distinct digits. We obtain all datapoints corresponding to the two digits of each pair from the MNIST dataset. Then, we create a $k'$-nearest-neighbor graph. We choose two very different values of $k'$, namely $k' = 300$ and $k' = 10$. The former value creates a dense nearest neighbor, resembling the regime of $O(n^2)$ edges, on which we optimize Objective 1. The latter values results in a sparse graph, resembling the regime with $O(n)$ edges, on which we optimize Objective 2 using our rule-of-thumb. The groundtruth here is defined by the MNIST labeling: there exist two latent features, each corresponding to whether a given image is the first or the second digit. Hence, each datapoint is labeled either as (01) or (10). Table 2 shows the results for $k' = 300$. The results for $k' = 10$ using our rule-of-thumb are qualitative the same. Indicatively, we report that for the pair of digits $(0, 7)$ precision and recall are $P = 0.9959$ and $R = 0.9821$ respectively. Similarly, for $(0, 8)$ we obtain $P = 0.9498, R = 0.8973$ respectively. A full evaluation of Objective 2 is shown in the next Section. As we see in Table 2 our method performs at least as well as BigClam. Furthermore, our method is typically at least 5 times faster than BigClam. It is worth outlining again that only BigClam among the competitors produced good quality results: CFinder produces poor quality results, verifying the claim of [44]. Finally, ILA does not complete after more than two days of running.

**Multilabel classification and robustness to noise.** For each of the Emotions and Flags datasets we construct a similarity graph by placing an edge between two entities if and only if they share some common feature. Our experiments address the following two questions: (a) Can we recover the true labeling using this similarity graph? (b) Is our method robust to noise? We model noise as random deletions of edges. Specifically, for each value of the sparsification parameter $p$, we delete each edge independently from the rest with probability $p$. We searched for the first value of $p$ for which either the precision or recall of our method is less than 90%. While $p$ is less than 0.1 we use a search step equal to 0.01 whereas for larger $p$ values we use a cruder search step of 0.025. Among the two resulting ranges we obtain, one for EMOTIONS and one for FLAGS, we use the superset for plotting purposes. Figure 4 presents the experimental results. We discuss them in detail in the following.

**EMOTIONS:** When $p = 0$, we perform 5 random experiments for our method, where the randomization is over the initial labelings $\omega_0$. For BigClam, we perform a single experiment. For each value $p > 0$ we perform 25 experiments for our method (5 random initial labelings $\times$ 5 sparsified graphs) and 5 for BigClam, one for each sparsified graph. All obtained averages are well concentrated around the mean.

We observe that for $p = 0.175$ our method achieves average precision $P = 0.91$ and recall $R = 0.93$ with an average running time equal to 32.83 seconds. For $p = 0.2$ we obtain $P = 0.61$ and $R = 0.67$ with an average running time 47.19 seconds. Hence the resulting range for $p$ is $\{0 : 0.01 : 0.1, 0.125 : 0.125 : 0.2\}$. Despite the fact that the total number of edges decreases as $p$ increases, the average running time -even if not strictly increasing- exhibits an increasing trend. BigClam even for $p = 0$ is unable to recover the ground-truth. Specifically, it achieves $P = 0.998$ and $R = 0.857$ with running time equal to 33.85 seconds. For values of $p$ up to 0.125 (included) the average precision is high $P \geq 0.867$ but the recall is extremely poor reaching the value $R = 0.026$. Above $p = 0.15$, precision drops as well. When $p = 0.20$, $P = 0.50$ and $R = 0.006$. The average speedup obtained by our method is 1.09. Notice that our method is not always faster than BigClam, specifically it is faster for 8 values of $p$.

**FLAGS:** We use the same number of experiments for each $p$ value as for EMOTIONS. Our method for $p = 0$ achieves $P = 0.986$ and $R = 0.996$ with an average running time 3.48 seconds. BigClam achieves $P = 0.999$ and $R = 0.995$ with average running time 8.66 seconds. Even if BigClam outperforms slightly our method in this case, as soon as $p = 0.01$, its average recall drops below 90%. Specifically, $P = 0.999$ and $R = 0.879$ with an average running time 9.1 seconds. Our proposed method up to $p = 0.15$ achieves concurrently precision and recall greater than 90%. Specifically for $p = 0.15$ our method achieves average precision $P = 0.95$ and average recall $R = 0.90$. The average running time is 8.75 seconds. For the next value $p = 0.175$, the average recall drops below 90%. Specifically, $P = 0.95$ and $R = 0.81$. The running time is 11.74 seconds. The average speedup achieved is 1.38 and our method is faster than BigClam for 10 values of $p$.

### 4.5 Learning social circles from ego-networks

In this Section we focus on a special type of induced graph. Specifically, we consider *ego-networks*, namely graphs induced by a vertex and its neighbors. Ego-networks play an increasingly important role in many applications. including user identification [21], anomaly detection [5] and viral marketing [26].

The application of interest to us was introduced by Leskovec et al. [29, 43, 44]. Consider a given social media user who connects to other users. We expect most of the connections -unless the user is a spammer- to be due to the fact that the two users share some common feature. For instance, it could be their common interest in sports, their common ethnic background or because they work in the same space. This intuition is reflected on the structure of these networks. As Yang and Leskovec point out, the probability of an edge is a non-decreasing function of their common features [44].

The question that Leskovec et al. consider is the following: *can we recover the social circles of a user in social media based on the structure of the network?*

Their answer is positive. We use our method to perform the same task. We compare our method to BigClam [43] on ego-networks from Twitter with ground-truth available. The number of latent features is set equal to the number of social circles. BigClam is a successful method for finding overlapping communities and detecting social circles [44]. The results are shown in Table 3. The first two columns provide a description of the dataset in terms of the number of vertices and edges respectively. As we see our method typically outperforms BigClam, both in terms of the quality and the time efficiency.

| Dataset | | FINLand | | | BigClam [43] | | |
|---|---|---|---|---|---|---|---|
| $n$ | $m$ | $P$ | $R$ | $t$ | $P$ | $R$ | $t$ |
| 227 | 3860 | 0.85 | 1.00 | 0.009 | 1.00 | 0.38 | 0.26 |
| 14 | 70 | 1.00 | 1.00 | 0.001 | 1.00 | 1.00 | 0.06 |
| 222 | 5871 | 0.90 | 0.96 | 0.074 | 0.97 | 0.61 | 0.26 |
| 150 | 2503 | 0.91 | 1.00 | 0.018 | 1.00 | 0.61 | 0.17 |
| 213 | 3508 | 1.00 | 1.00 | 0.001 | 1.00 | 1.00 | 0.14 |
| 85 | 547 | 0.91 | 0.93 | 0.025 | 1.00 | 0.49 | 0.03 |
| 33 | 235 | 1.00 | 0.89 | 0.002 | 0.92 | 0.96 | 0.11 |
| 34 | 284 | 1.00 | 1.00 | 0.002 | 1.00 | 0.30 | 0.08 |
| 192 | 4190 | 0.89 | 1.00 | 0.054 | 1.00 | 0.55 | 0.05 |
| 120 | 577 | 1.00 | 0.92 | 0.002 | 1.00 | 0.55 | 0.37 |
| 99 | 1519 | 0.85 | 0.91 | 0.006 | 1.00 | 0.39 | 0.06 |
| 225 | 2602 | 1.00 | 0.96 | 0.018 | 0.96 | 0.75 | 0.19 |
| 56 | 665 | 0.87 | 0.96 | 0.038 | 0.96 | 0.39 | 0.30 |
| 113 | 1689 | 0.86 | 0.92 | 0.018 | 0.93 | 0.50 | 0.22 |
| 58 | 479 | 1.00 | 1.00 | 0.002 | 1.00 | 1.00 | 0.16 |

**Table 3: Learning social circles from ego-networks from Twitter results. For details, see text of Section 4.5.**

## 5. RELATED WORK

**Learning Latent Features.** The infinite relational model (IRM) is a prominent method among latent class models [23]. It assumes that each entity belongs to a single cluster or -equivalently in our terminology- it possesses a single feature. Edge probabilities depend solely on cluster assignments. Notice that this is reminiscent of McSherry's planted partition model [31]. However contrary to the planted partition model, IRM makes additional probabilistic assumptions. Specifically, it assumes that cluster assignments are drawn from the Chinese Restaurant Process [6]. In general, latent class models can model multiple participation to features, at the cost of combinatorial explosion: for each possible combination of features, create a new feature. Latent feature models allow multiple participation to features without the aforementioned cost [32]. Recently, Palla et al. [37] introduced the *infinite latent attribute* (ILA) model which yielded improvements over both [23] and [32]. Despite the accurate performance of ILA on small datasets, the method does not scale even to graphs with few hundreds of edges.

**Overlapping clustering.** Leskovec et al. showed that major non-overlapping clustering methods typically produce meaningful communities when their size is restricted to few tenths of vertices [28]. Recently, Yang and Leskovec show that overlapping communities can explain the core-periphery structure in networks [44]. Khandekar, Kortsarz and Mirrokni mathematically formalize this phenomenon and prove that overlapping clustering may yield significant benefits compared to the non-overlapping case [24]. A solid modeling approach to providing a theoretical model of communities that is also able to reproduce certain salient characteristics of real world networks is the affiliation networks model [27]. This model lies conceptually close to many research works including ours: the latent labelings our method produces can be seen as a bipartite graph where each vertex is afilliated to each feature it posseses. Anandkumar et al. provide guaranteed community recovery for the mixed membership Dirichlet model [4]. Gopalan and Blei use the same model [19] and probabilistic inference algorithms to detect overlapping communities. Arora et al. [9] can detect overlapping communities under other modeling assumptions. CFinder is a popular method for detecting overlapping communities using clique percolation [3]. CFinder does not scale well to large networks and more importantly it fails to find overlapping communities even in the context of two communities with a dense overlap. As Yang and Leskovec frequently this is the case on social networks [44]. It is worth mentioning that in the extreme case when the overlaps are complete, then it is known how to find these clique-separators in polynomial time [39]. However, the overlaps tend to be large near-cliques [40] rather than perfect cliques. Mcauley and Leskovec use overlapping community detection methods to discover social circles from ego-networks [29]. A subsequent method developed by Yang and Leskovec [43] is based on non-negative matrix factorization and scales better than [29]. As we saw in Section 4 our method outperforms [43], is significantly simpler and comes with solid theoretical guarantees. An interesting research direction is to understand the effects of recent developments in nonnegative matrix factorization [8] for detecting overlapping communities, since the underlying assumptions of Arora, Ge, Kannan and Moitra are -in many cases- realistic. Closely related to non-negative matrix factorization approaches is the geometric approach proposed in [41]. The concept of $(\alpha, \beta)$-clusters is interesting within the context of overlapping communities but recent developments have shown that under a well-accepted conjecture finding even one $(\alpha, \beta)$-cluster is intractable [10]. Finally, many other heuristics exist that are not well understood exist, e.g., [2, 14, 18, 35, 45]. As we discussed earlier our work sheds light on the heuristic method developed by Bonchi et al. [14] .

## 6. CONCLUSION

**Summary.** In this work we consider a fundamental problem with many important applications: can we learn efficiently and accurately latent features that explain a given graph? We propose a simple probabilistic model and a rapidly mixing Metropolis chain for optimizing the resulting maximum likelihood function. We use our method to perform successfully multilabel classification from similarity graphs and to learn social circles from Twitter ego-networks.

**Open Problems.** Our work opens numerous questions. What are the graph sub-structures that affect the performance of latent feature learning methods? Under our model, we showed that complete bipartite subgraphs play a key role. What about other models such as the ILA [37]? Combining such answers with the work of Ugander et al. [42] may shed light on the merits and limitations of such methods on real-world networks. Why does the simple probabilistic model we introduce perform so well on real world datasets? How do we include non-assortative features? Can we have a richer family of probabilistic models for undirected graphs for which we can have efficient (approximation) algorithms? Developing new probabilistic models and latent feature learning algorithms for directed graphs is also an interesting research direction. Finally, in future work we plan to use our method on other graph mining problems such as link prediction.

## 7. REFERENCES

[1] http://yann.lecun.com/exdb/mnist/. 4.1

[2] *Evolutionary algorithms for overlapping correlation clustering*, 2014. 5

[3] B. Adamcsek, G. Palla, I. J. Farkas, I. Derényi, and T. Vicsek. CFinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22(8):1021–1023, 2006. 4.1, 4.2, 5

[4] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. In *NIPS'09*, pages 33–40, 2009. 5

[5] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *Advances in Knowledge Discovery and Data Mining*, pages 410–421. Springer, 2010. 4.5

[6] D. J. Aldous. *Exchangeability and related topics.* Springer, 1985. 5

[7] R. Andersen, D. F. Gleich, and V. Mirrokni. Overlapping clusters for distributed computation. In *WSDM*, pages 273–282. ACM, 2012. 1

[8] S. Arora, R. Ge, R. Kannan, and A. Moitra. Computing a nonnegative matrix factorization–provably. In *STOC*, pages 145–162. ACM, 2012. 5

[9] S. Arora, R. Ge, S. Sachdeva, and G. Schoenebeck. Finding overlapping communities in social networks: toward a rigorous approach. In *ACM EC*, pages 37–54. ACM, 2012. 5

[10] M.-F. Balcan, C. Borgs, M. Braverman, J. T. Chayes, and S.-H. Teng. Finding endogenously formed communities. In *SODA '13*, pages 767–783, 2013. 5

[11] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1–3):89–113, 2004. 1

[12] E. Becker, B. Robisson, C. E. Chapple, A. Guénoche, and C. Brun. Multifunctional proteins revealed by overlapping clustering in protein interaction network. *Bioinformatics*, 28(1):84–90, 2012. 1

[13] P. Boldi and S. Vigna. The webgraph framework i: compression techniques. In *WWW'04*, pages 595–602, 2004. 1

[14] F. Bonchi, A. Gionis, and A. Ukkonen. Overlapping correlation clustering. In *ICDM '11*, pages 51–60, 2011. 1, 3, 1, 3, 5

[15] R. Bubley and M. Dyer. Path coupling: A technique for proving rapid mixing in markov chains. In *FOCS*, pages 223–231. IEEE, 1997. 1

[16] F. Doshi-Velez and Z. Ghahramani. Accelerated

sampling for the indian buffet process. In *ICML'09*, pages 273–280, 2009. 4.2

[17] F. Doshi-Velez and Z. Ghahramani. Correlated non-parametric latent feature models. In *UAI*, pages 143–150, 2009. 4.2

[18] T. Evans and R. Lambiotte. Line graphs, link partitions, and overlapping communities. *Physical Review E*, 80(1):016105, 2009. 5

[19] P. K. Gopalan and D. M. Blei. Efficient discovery of overlapping communities in massive networks. *Proceedings of the National Academy of Sciences*, 110(36):14534–14539, 2013. 5

[20] T. Griffiths and Z. Ghahramani. Infinite latent feature models and the indian buffet process. *TR 2005-001, Gatsby Comp. Neuroscience Unit.*, 2005. 4.2

[21] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos. It's who you know: graph mining using recursive structural features. In *KDD*, pages 663–671. ACM, 2011. 4.5

[22] P. F. Jonsson and P. A. Bates. Global topological features of cancer proteins in the human interactome. *Bioinformatics*, 22(18):2291–2297, 2006. 1

[23] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *AAAI'06*, pages 381–388, 2006. 1, 5

[24] R. Khandekar, G. Kortsarz, and V. Mirrokni. On the advantage of overlapping clusters for minimizing conductance. *Algorithmica*, 69(4):844–863, 2014. 1, 5

[25] S. Kirkpatrick and C. D. Gelatt and MP. Vecchi. Optimization by simulated annealing In *Science*, 1983. 4.2

[26] S. Lattanzi and Y. Singer. The power of random neighbors in social networks. In *WSDM*, 2015. 4.5

[27] S. Lattanzi and D. Sivakumar. Affiliation networks. In *STOC*, pages 427–434, 2009. 5

[28] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW*, pages 695–704. ACM, 2008. 5

[29] J. Leskovec and J. J. Mcauley. Learning to discover social circles in ego networks. In *NIPS*, pages 539–547, 2012. 1, 4.1, 4.5, 5

[30] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov chains and mixing times*. American Mathematical Soc., 2009. 2, 3

[31] F. McSherry. Spectral partitioning of random graphs. In *FOCS*, pages 529–537. IEEE, 2001. 5

[32] K. Miller, M. I. Jordan, and T. L. Griffiths. Nonparametric latent feature models for link prediction. In *NIPS*, pages 1276–1284, 2009. 1, 1, 5

[33] B. Morris, W. Ning and Y. Peres. Mixing time of the card-cyclic-to-random shuffle. The Annals of Applied Probability, 24:5, pages 1835–1849, 2014 4.1

[34] D. Mumford. Pattern theory: a unifying perspective. In *First European congress of mathematics*, pages 187–224. Springer, 1994. 1

[35] T. Nepusz, A. Petróczi, L. Négyessy, and F. Bazsó. Fuzzy communities and the concept of bridgeness in complex networks. *Physical Review E*, 77(1):016107, 2008. 5

[36] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005. 1

[37] K. Palla, D. A. Knowles, and Z. Ghahramani. An infinite latent attribute model for network data. In *ICML*. 2012. (document), 1, 4.1, 4.2, 4.2, 5, 6

[38] I. Psorakis, S. Roberts, M. Ebden, and B. Sheldon. Overlapping community detection using bayesian non-negative matrix factorization. *Physical Review E*, 83(6):066114, 2011. 1

[39] R. E. Tarjan. Decomposition by clique separators. *Discrete mathematics*, 55(2):221–232, 1985. 5

[40] C. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *KDD*, pages 104–112. ACM, 2013. 5

[41] C. E. Tsourakakis. Toward quantifying vertex similarity in networks. *Internet Mathematics*, 10(3-4):263–286, 2014. 5

[42] J. Ugander, L. Backstrom, and J. Kleinberg. Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections. In *WWW*, pages 1307–1318, 2013. 6

[43] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *WSDM*, pages 587–596. ACM, 2013. (document), 1, 4.1, 4.2, 4.2, 4.2, 4.3, 4.5, 5

[44] J. Yang and J. Leskovec. Overlapping communities explain core-periphery organization of networks. Technical report, Stanford University, October 2014. (document), 1, 4.2, 4.3, 4.4, 4.5, 5

[45] Z.-Y. Zhang, Y. Wang, and Y.-Y. Ahn. Overlapping community detection in complex networks using symmetric binary matrix factorization. *Physical Review E*, 87(6):062803, 2013. 5