A Bottom-up Hierarchical Approach for Joint Inference of Overlapping Social Circles and Missing User Information

Siddharth Kannan
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
siddhark@andrew.cmu.edu

Vignesh Kannan
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
vkannan2@andrew.cmu.edu

Abstract

With the massive amount of information generated by social networks everyday, it is becoming increasingly important to organize this data. One way to do so is by grouping them into social circles (a group of people having some common attributes). In this project, we address the problem of automatically discovering social circles in a network induced by a user and his friends called an ego-network. We propose a novel hierarchical, non-parametric approach to solving this problem which exploits both the structural information as well as the user-attribute information using the principle of homophily. Our method addresses one of the major drawbacks of hierarchical clustering that it does not allow the formation of overlapping clusters. Another strength of our algorithm is that it does not require the a priori information of the number of social circles. Results on the Facebook dataset of ego-networks show that our method performs on comparable terms to the benchmark achieved by (1).

1 Introduction

Social networking sites have transformed information sharing and connectivity between people in the last decade. With the emergence of various platforms like Facebook, Twitter and Google+ there has been an explosion of data associated with these social networks and an increasing interest to model them. A common building block of such large social networks is a *social circle*. Put in simple terms, a social circle is a group of users sharing some common attributes like the same university or the same residential location and so on. It is natural to think of social networks as being composed of many interconnected social circles and indeed, the automated discovery of these latent circles has been an active area of research and will be the focus of this project.

These social circles offer a lot of valuable information and insight for several applications. Users can choose to filter sharing of content to only a select few social circles out of all the social circles they are a part of. This ensures that friends only see information which is intended for them. Another application is strategic marketing of products. Organizations can target advertisements to groups or circles which are most likely to use a specific product rather than broadcasting it. From the perspective of international security, we can use the structural information of these social circles to identify anomalies and potential terrorist connections. Additionally, we can also infer missing information about users from their memberships to different social circles and improve friend and page suggestions to them. This can also facilitate the formation of online communities and forums to mobilize support for various causes. Finally, the study of the formation of these social circles can be of significant importance from the standpoint of social sciences too as we can gather new insights

Preprint. Work in progress.

about the interconnected and interwoven nature of society itself. Hence this problem has a lot of relevance in today's scenario.

We formally define our problem as follows. We are given a node (user) i referred to as the *ego*, and his set of friends $\mathcal{F}_i$. We are also given the connections between his friends in the form of edges $\mathcal{E}$. Using this information, we define the graph induced by the ego i and his friends $\mathcal{F}_i$ (which includes $\mathcal{E}$) as the *ego-network* $\mathcal{G}$. Our aim is to automatically discover a set of social circles $\mathcal{C} = \{C : C \subset j \cup \mathcal{F}_j\}$ in this graph $\mathcal{G}$ such that these circles have nodes which are similar to each other with respect to some set of attributes. Below we clearly state the properties which we would like our algorithm to capture in order to effectively solve this problem.

- Social circles evolve in a hierarchical bottom-up fashion.
- Social circles can overlap i.e. a user can belong to more than one circle.
- Smaller social circles can form within bigger social circles i.e. hierarchical circles exist.
- The principle of *homophily* needs to be captured by our algorithm.
- Social circles are dynamic in nature as they constantly get modified by the addition of new friends to a user or change in attributes of the user.
- Users with many mutual friends are more likely to be connected.

2 Related Work

(1) models the probability of existence of an edge between two vertices. They treat the membership of vertices to the circles as latent variables and propose an Expectation Maximization (EM) like framework to alternately optimize circle membership and the parameters of the user profile similarity function until convergence. This enables the incorporation of both structural and node content information in the model. This is a parametric approach where in the parameters are used to encode the dimensions of profile similarity due to which circles were formed. Hence this makes it an interpretable method in the sense that one can trace the attributes which caused a circle to emerge. Another important aspect of this method is that it needs the number of circles to be provided beforehand. The authors choose the number of clusters by minimizing an approximation to the Bayesian Information Criterion (BIC). We use this method as the benchmark for our approach since we are concerned with the same problem setting. Also, we borrow the evaluation methods and metrics used by the authors in order to compare our results with that of their approach.

The second approach operates from a global perspective wherein we are interested in modeling social circles on the whole graph. To this end, (2) proposes an approach that uses a stacked graph autoencoder to combine node content and graph structure in the spectral domain and performs a spectral clustering on the learned representation. Additionally, they add random noise to the node content in order to capture the dynamic interplay between node content and structure and are also able to achieve a closed form solution for the autoencoder.

Another approach is proposed by (3) wherein the main focus is to address the problem of missing information which could either be missing links between two users or missing attributes of a user. The solution here is based on the *homophily* phenomenon which states that users with similar attributes tend to form links between each other and vice versa. We borrow this principle to incorporate structural information as well as user attribute information in our clustering algorithm. In a unified probabilistic framework, the authors propose an iterative algorithm composed of two alternating steps: Graph Construction (GC) and Label Propagation (LP). GC tries to estimate the probability of link formation between pairs of users given some already existing links and the attribute information of users. LP tries to infer attributes for users based on some already given attributes and probability of link formation (affinity) between pairs of users. The two steps alternate with each other until convergence. Our algorithm uses a similar framework alternating between Graph Construction and Label Propagation but the computation performed at each step differs significantly from the work of (3). Also, in (3) there is no provision for explicitly modeling social circles, rather, they only focus on completing missing links and missing attributes. But we use the principle of homophily to discover social circles.

(4) proposes to tackle the discovery problem using a hierarchical Gaussian Mixture Model (GMM), but this makes it dependent on supplying the number of communities (K) beforehand which is not

feasible when there are millions of nodes. Also, the strong Gaussian assumptions may not be true in a real-world setting.

3 Methodology

3.1 Approach

Based on the properties stated in the Introduction section, we outline our approach below.

Algorithm 1

```
1:  procedure MAIN($\theta, \alpha, T$)
2:      Number the friends of the given ego from 1 to n
3:      Define $\mathcal{C} = \{c_i : 1 \leq i \leq n\}$
4:      Define $\mathcal{V} = \{1, 2, ....n\}, \mathcal{E} = \{(i, j)|$ i and j are friends$\}$
5:      for $c_i \in \mathcal{C}$ do
6:          $c_i.members = i$
7:      $t = 0$
8:      while $t \leq T$ do
9:          SORTDESC($\mathcal{E}$)            ▷ edges in $\mathcal{E}$ are sorted in descending order of similarity
10:         CIRCLEFORMATION($\mathcal{E}$)
11:         MERGECIRCLES($\mathcal{C}, \theta$)
12:         LABELPROPAGATION($\mathcal{V}, \alpha$)
13:         Define $\mathcal{J} = \frac{1}{n}\sum_{i=1}^{n} ||i.attributes^{(t+1)} - i.attributes^{(t)}||_1$
14:         Terminate if $\mathcal{J} < \epsilon$
15:         $t = t + 1$
16:
17: procedure CIRCLEFORMATION($\mathcal{E}$)
18:     for $(i, j) \in \mathcal{E}$ do
19:         for $c \in i.membership \setminus j.membership$ do
20:             ADDTOCIRCLE($c$, $j$)
21:         for $c \in j.membership \setminus i.membership$ do
22:             ADDTOCIRCLE($c$, $i$)
23:
24: procedure MERGECIRCLES($\mathcal{C}, \theta$)
25:     for $c_i \in \mathcal{C}$ do
26:         for $c_j \in \mathcal{C}$ do
27:             if IOU($c_i, c_j$) $> \theta$ then
28:                 for $k \in c_j.members$ do
29:                     Remove $c_j$ from $k.membership$
30:                     Add $c_i$ to $k.membership$
31:                     Add $k$ to $c_i.members$
32:                 Delete $c_j$ from $\mathcal{C}$
33:
34: procedure LABELPROPAGATION($\mathcal{V}, \alpha$)
35:     for $i \in \mathcal{V}$ do
36:         $neighbors = \{j \in c.members : c \in i.membership\}$
37:         $i.attributes_{new} = \sum_{k \in neighbors} w_{ki} \cdot k.attributes$
38:         $i.attributes = \alpha \cdot i.attributes + (1 - \alpha) \cdot i.attributes_{new}$
39:
40: procedure ADDTOCIRCLE($c$, $i$)
41:     if $aggregate\_similarity(c, i) > 0.5$ then
42:         Add $c$ to $i.membership$
43:         Add $i$ to $c.members$
```

3.1.1 Initialization

In this step, each node is assigned to be its own circle. This is similar to the base case in a social network when there are no connections. Every user is just an isolated node. We then initialize each edge weight as the similarity between the two nodes forming the edge. Computing the similarity is crucial to our algorithm since the principle of homophily heavily relies on it. Two similarity measures have been explored in our algorithm. One is the cosine-similarity while the other is the Gaussian kernel. Mathematically, we define them as follows:

$$similarity(i,j) = \begin{cases} \frac{\langle i.attributes, j.attributes \rangle}{||i.attributes||_2 ||j.attributes||_2}, & \text{if using cosine-similarity} \\ \exp\Big(- \frac{||i.attributes - j.attributes||_2^2}{\sigma^2} \Big), & \text{if using Gaussian kernel} \end{cases}$$

where $i.attributes$ and $j.attributes$ denote the attribute vectors of i and j respectively. From the above definition, it is clear that the similarity between two nodes i and j ranges from 0 to 1, with 0 representing no similarity and 1 representing complete similarity.

3.1.2 Edge Sorting

Following initialization we sort the edges in the decreasing order of weight (similarity). The rationale behind this step is borrowed from Kruskal's algorithm for Minimum Spanning Trees. But for our use case we are concerned with maximizing the similarity between nodes of social circle, so we reverse-sort the edges. The reason for sorting the edges is that we want social circles to start forming from pairs of nodes which are very similar. This would help us capture the real-world setting where in people who share a lot of similarities tend to be the closest to each other and hence are the first to form mutual connections on social media. Moreover, this is precisely how hierarchical clustering starts forming clusters starting from nodes which are the most similar to each other.

3.1.3 Circle Formation

We then consider each edge in the sorted order and try growing bigger circles out of the existing circles. Using edges to guide the growing of circles helps us leverage the structural information of the ego-network. In comparison to naive hierarchical clustering which only takes into account similarity between nodes while growing clusters, our method builds circles which are supported by a framework or skeleton of edges. While this might be disadvantageous in the case of missing links in he dataset we argue that our subsequent *Label Propagation* step will help us propagate attribute information across a circle thus allowing two nodes to be put into the same circle even if they don't have a direct edge connecting them by virtue of their neighboring nodes.

Now we state the processing we do for each edge in order to grow the circles. Let's consider an edge (i, j). We try adding node i to every circle of node j which node i is not already a part of. We also try adding node j to every circle of node i which node j is not already a part of. This particular step is pivotal in the algorithm being able to achieve overlapping circles. Essentially, it allows nodes to have multiple circle memberships. As the algorithm progresses, we see that each user node gets a chance to be a part increasingly many circles limited by the similarity of the node to the circle under consideration.

Similar to hierarchical clustering, to add a node to a circle, we need to define a notion of aggregate similarity between a node and a circle. We try out both Complete-link and Average-link techniques of hierarchical clustering for this purpose. More precisely the aggregate similarity between node i and circle c is defined as,

$$aggregate\_similarity(c,i) = \begin{cases} \min_{j \in c.members} \; similarity(i,j), & \text{if using Complete-link} \\ \text{avg}_{j \in c.members} \; similarity(i,j), & \text{if using Average-link} \end{cases}$$

If the aggregate similarity of node i with circle c is greater than 0.5, we add node i to the members of c and add c to the membership of i. Otherwise, there is no change with respect to node i and circle c. This step terminates after processing all the edges of the ego-network $\mathcal{G}$. The output is a set of overlapping, hierarchical and disjoint circles.

3.1.4 Merge Circles

While the previous step results in different kind (overlapping, hierarchical, and disjoint) of circles as desired, it does no eliminate redundant circles. As is evident from the initialization, we start of with each node as it's own cluster and the *Circle Formation* step does nothing to reduce the number of circles. It is also easy to see that we end up with many redundant circles from the previous step, some are even exactly the same in the first few iterations of the algorithm. Two circles c_i and c_j are considered to be redundant if they have a large overlap between their respective members. To quantify this overlap, the *Intersection over Union (IoU)* metric is used. The IoU score for a pair of circles c_i and c_j is computed as follows,

$$IoU(c_i, c_j) = \frac{c_i.members \cap c_j.members}{c_i.members \cup c_j.members}$$

Putting it together, we consider all possible pairs of circles c_i and c_j, compute their IoU score $IoU(c_i, c_j)$ and check if it is above a certain threshold θ. If $IoU(c_i, c_j) > \theta$ we deduce that c_i and c_j have a large overlap and are essentially duplicates of the same circle. To merge c_i and c_j into a single circle, we transfer all the members of c_j to c_i thus updating $c_i.members$ and $k.membership : k \in c_j.members$. After the transfer, c_j is deleted and does not participate in further iterations. Note that in the above merging step, it doesn't matter if we transfer the members of c_j to c_i and delete c_j or vice-versa. The resulting merged circle will be the same either way. Only the circle index may differ, but this has no bearing on the performance of the algorithm. The range of $IoU(c_i, c_j)$ (and hence θ) is from 0 to 1 where 0 indicates no overlap between c_i and c_j, and 1 indicates that c_i and c_j are identical. This metric does most of the trick for us in capturing overlapping, disjoint and hierarchical circles. A small IoU score helps us retain such circles rather than merging them. We discuss how we choose θ in the Experiments and Results section.

3.1.5 Label Propagation

There have been many efforts to infer missing labels of unlabeled data by propagating the labels of labeled data. Some approaches use soft label assignments while others opt for hard assignment. Popular approaches are proposed by [(3), (5), (6), (7)]. Some of them like (6) refer to class labels as "labels" while others like (3) use the term "labels" to denote attributes of nodes. We use the latter notation. But the methods of propagation in either case are closely related. The purpose of *Label Propagation* is to capture the idea that people who are connected (or are part of the same social circle) tend to have similar attributes.

In this step, we consider each node i and update its attributes based on the attributes of its *neighbors*. The set of *neighbors* is defined as follows,

$$neighbors = \{j \in c.members : c \in i.membership\}$$

The update computes a weighted average of the attributes of the neighbors. The weight given to a neighbor is proportional to the similarity of the neighbor to the node i. Mathematically, the update is as shown below,

$$i.attributes_{new} = \sum_{k \in neighbors} w_{ki} \cdot k.attributes$$

$$i.attributes = \alpha \cdot i.attributes + (1 - \alpha) \cdot i.attributes_{new}$$

where w_{ki} is defined as,

$$w_{ki} = \frac{similarity(k, i)}{\sum_{j \in neighbors} similarity(j, i)}$$

The hyperparameter α is used to control the extent of the update. In some sense, it computes a moving average of the updates over each iteration of the algorithm. Lower the value of α, more aggressive is the update. Aggressive updates encourage the formation of large social circles.

3.1.6 Convergence

To check for convergence, we use the update performed by *Label Propagation* to see how much the attributes of nodes have changed. If the change is small, we stop the algorithm and declare convergence. More precisely, we compute the $l1$-norm of the difference between the old attributes and the updated attributes of each node as shown below,

$$\mathcal{J} = \frac{1}{n}\sum_{i=1}^{n} ||i.attributes^{(t+1)} - i.attributes^{(t)}||_1$$

where $i.attributes^{(t+1)}$ is the updated attribute vector for node i and $i.attributes^{(t)}$ is the old attribute vector of node i. If $\mathcal{J} < \epsilon$, convergence is declared and we terminate. Otherwise, we repeat the whole process again in the next iteration.

3.2 Time Complexity

Let $|max\_members|$ be the maximum number of users belonging to a social circle. Let $|max\_membership|$ be the maximum number of circles that a user belongs to. The time-complexity of the proposed algorithm is given as follows,

$$\text{Time Complexity: } O\Big(T\Big(E\log E + V^2(|max\_members| + |max\_membership|) + E|max\_membership||max\_members|\Big)\Big)$$

4 Dataset Description

We experiment our algorithm on the Facebook Social Circles dataset (8). It consists of 4039 nodes connected by 88234 edges with an average clustering coefficient of 0.6055 (clustering coefficient of node is a measure of how close its neighbors are to forming a clique). The attributes for each node are represented in binary format while obscuring the true interpretation of the attribute's value. For example, gender is encoded as 2 bits, 1 for each gender but we do not have information about which bit represents which gender. There are a total of 10 ego-networks in this dataset. As ground truth, the dataset also provides a list of social circles for every ego-network. Each circle consists of a list of members IDs belonging to that circle. These ground truth social circles were collected from participants in a survey using the Facebook app.

5 Experiments and Results

5.1 Performance Evaluation

We use the same evaluation metrics as (1). The first metric is the BER_score and the other is the $F1\_score$. BER (Balanced Error Rate) for a binary classification problem is given by,

$$BER = \frac{1}{2}\Big(\frac{FN}{TP+FN} + \frac{FP}{TN+FP}\Big)$$

For our case, given a predicted circle C_p and a ground-truth circle C_g we can treat this a binary classification problem. The nodes which fall into the circle C_p can be treated as examples which are classified as 1, and the rest that fall outside C_p can be treated as examples which are classified as 0. Similarly, nodes which fall inside C_g are treated as 1's in the ground-truth and the rest which fall outside C_g are treated as 0's in the ground-truth. The BER for this classification problem can then be defined as,

$$BER(C_p, C_g) = \frac{1}{2}\Big(\frac{|C_g \setminus C_p|}{|C_g|} + \frac{|C_p \setminus C_g|}{|C_g^c|}\Big)$$

Similarly, we can also compute the precision and recall values for this classification problem as follows,

$$precision(C_p, C_g) = \frac{|C_p \cap C_g|}{|C_p|}$$

$$recall(C_p, C_g) = \frac{|C_p \cap C_g|}{|C_g|}$$

The $F1(C_p, C_g)$ value can then be computed using these values.
Just as in (1), we align the ground-truth circles with the predicted circles by solving the linear assignment problem. Using the computed assignment, we calculate the BER_score and the $F1\_score$ as follows,

$$BER\_score(\mathcal{C}_p, \mathcal{C}_g) = \max_{f \in \mathcal{F}} \frac{1}{|dom(f)|} \sum_{C_p \in dom(f)} BER\_score(C_p, f(C_p))$$
$$F1\_score(\mathcal{C}_p, \mathcal{C}_g) = \max_{f \in \mathcal{F}} \frac{1}{|dom(f)|} \sum_{C_p \in dom(f)} F1\_score(C_p, f(C_p))$$

where $\mathcal{F} = \{f : \mathcal{C}_p \to \mathcal{C}_g\}$ is the family of all linear assignment functions from $\mathcal{C}_p$ (set of predicted circles) to $\mathcal{C}_g$ (set of ground-truth circles). As per the assumption in (1), we do not penalize extra predictions. The number of circles predicted in our algorithm is significantly far from the trivial solution of the power set of all nodes. In fact, for some of the ego-networks, we predict number of circles close to the number of circles in the ground truth. Also, our algorithm tries to infer the number of circles which is quite a challenging task and one of its merits. So it is reasonable to not harshly penalize the model for getting the number of circles wrong in some cases.

5.2 Experiments

We conducted all our experiments on the Stanford Network Analysis Project (SNAP), Social circles: Facebook dataset described previously. The algorithm was run on each of the 10 ego-networks separately and the retrieved social circles were compared with the ground truth circles to evaluate the performance of the algorithm.

We experimented with different combinations of settings for the similarity measure between nodes (using cosine-similarity and the Gaussian kernel) and the aggregate similarity between a node and a circle (using Complete-link or Average-link). We report the results for all these 4 combinations in the following section.

In our experiments, we chose the value of $\theta = 0.6$. This was based on a trade-off between the precision and the recall of the predicted circles. We observed that setting θ to a low value resulted in very few circles each of which was undesirably large. This caused low recall values and relatively higher precision values. Using a high value for θ had the opposite effect. It resulted in too many undesirably small circles. Thus the recall improved, but the precision dropped. So we achieved a trade-off between the two quantities by choosing $\theta = 0.6$. Moreover, we noticed that this value of θ has further justification with respect to this dataset since the average clustering index is close to 0.6 and the clustering index captures a similar notion as IoU for edges. Because our *Circle Formation* step is guided by edges, it is reasonable to expect the ideal θ value to be close to the average clustering index.

We used $\alpha = 0.99$. This was again based on our observations from multiple trials. Choosing an α value any lower than 0.99 made the clustering very aggressive, thus resulting in very few circles each with a large size. An explanation for this is that, as we proceed with many iterations, the value of α decays exponentially since we are multiplying α at each stage. So even if we run the algorithm for around 80 iterations, the resultant α would be $0.99^{80} \approx 0.45$.

For each ego-network we specified the maximum number of iterations as 80. From our observation, the algorithm converged before this mark for some of the ego-networks while others ran till 80 iterations.

5.3 Results

The below table summarizes the results obtained from our experiments. We report the average of the scores obtained from the 10 ego-networks for each of the 4 combinations of settings described previously.

Method		Average BER_score	Average $F1\_score$
Link Type	Similarity		
Average Link	Cosine	0.6207	0.2775
Complete Link	Cosine	0.6898	0.3634
Average Link	Gaussian Kernel	0.6223	0.4247
Complete Link	Gaussian Kernel	**0.7158**	**0.4667**
McAuley et al. 2012		0.8400	0.5900

Table 1: Results of 4 combinations of settings

5.4 Interpretation and Discussion of Results

As suggested by the results, we see that the setting with the Gaussian kernel for similarity and Complete-link for aggregate similarity outperforms the other settings with an average BER_score of 0.7158 and an average $F1\_score$ of 0.4667. A more refined observation is that Complete-link outperforms Average-link. Keeping the similarity measure the same, using Complete-link over Average-link gives a 0.09 increase in average BER_score for the Gaussian kernel and a 0.07 increase in average BER_score for cosine-similarity. Similarly, we get a 0.05 increase in average $F1\_score$ for the Gaussian kernel and a 0.09 increase in average $F1\_score$ for cosine-similarity.

This trend can be explained as follows. Average-link in general tends to be more aggressive in forming bigger circles when compared to Complete-link. The reason for this is that Average-link uses the average operation to compute aggregate similarity of node to a circle. When using average, the aggregate similarity gets boosted by nodes in the circle which are similar to the node under consideration. Hence there is a higher chance that the node is assigned to the circle. But using Complete-link makes the criterion for a node to be added to a circle more rigid. It dictates that a node is added to a circle only if the all the nodes (captured by minimum similarity) in the circle are similar to the node under consideration. So this prevents the algorithm from aggressively clustering many nodes into a single circle. This also benefits the precision of each predicted circle since we naturally have smaller circles with Complete-link and hence a better BER_score and $F1\_score$.

Another point of comparison is that the Gaussian kernel tends to give a higher $F1\_score$ than cosine-similarity. For Complete-link, using the Gaussian kernel over cosine-similarity increases the average $F1\_score$ by 0.1, where as, for Average-link the magnitude of increase is even higher at 0.15. Finally, we see that our approach has a performance comparable to that of the benchmark, falling short by 0.12 on each of the average BER_score and the average $F1\_score$. We believe that the future work mentioned in the following section can significantly help reduce this gap and hence is promising direction to work in.

6 Conclusion and Future Work

We have presented a novel algorithm for the inference of social circles and potentially missing user information from a given ego-network. Our algorithm is able to capture all three kinds of social circles that can form, namely:- partially overlapping, hierarchical and disjoint. Additionally, we overcome the need to have the number of circles to detect beforehand. As future work, we want to establish the interpretability of the social circles we have generated. A simple approach is to analyze the circles for the attribute which has the least variance among the rest within a given circle. We hope to formally prove the convergence of our algorithm and analyze the relationship between the number of iterations required for convergence and the size of the ego-network. We noticed in the dataset that some of the features like gender are one-hot encodings while others like education degree are not. Accounting for this in the *Label Propagation* step can help improve performance. Finally we plan to evalutate our algorithm on larger datasets like Google+ and Twitter for which improving the time complexity of the algorithm may be necessary.

References

[1] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *Advances in neural information processing systems*, pp. 539–547, 2012.

[2] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "Mgae: Marginalized graph autoencoder for graph clustering," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 889–898, ACM, 2017.

[3] C. Yang, L. Zhong, L.-J. Li, and L. Jie, "Bi-directional joint inference for user links and attributes on large social graphs," in *Proceedings of the 26th International Conference on World Wide Web Companion*, pp. 564–573, International World Wide Web Conferences Steering Committee, 2017.

[4] H. Zhang, C. L. Giles, H. C. Foley, and J. Yen, "Probabilistic community discovery using hierarchical latent gaussian mixture model," in *AAAI*, vol. 7, pp. 663–668, 2007.

[5] D. Chakrabarti, S. Funiak, J. Chang, and S. A. Macskassy, "Joint inference of multiple label types in large networks," *arXiv preprint arXiv:1401.7709*, 2014.

[6] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," 2002.

[7] M. Sattari and K. Zamanifar, "A spreading activation-based label propagation algorithm for overlapping community detection in dynamic social networks," *Data & Knowledge Engineering*, vol. 113, pp. 155–170, 2018.

[8] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection." `http://snap.stanford.edu/data`, June 2014.