A Survey on Community Mining in Social Networks

M. Jalal*, A. Doan*
* Department of Computer Science, University of Wisconsin-Madison
{jalal, anhai }@cs.wisc.edu

I. Abstract

Community detection and overlapping community detection has been of significant importance in the last decade wherein invention and growth of social networks like Facebook, Twitter, Linkedin, Flickr, etc. has even made it more crucial to investigate better approaches. Overlapping community detection has many application as in modern market analysis and recommendation systems, bioinformatic systems, etc. The main hindrance in analysing social networks like Facebook is their giant social graphs and problems like detecting communities and their overlaps have very large time complexity. Finding overlapping communities in modern social graphs is an open problem and researchers have been using many heuristic graph mining and machine learning algorithms to approach the problem with less complexity. In this study we survey various community detection algorithm used in the literature as well as methods for evaluating them.

Keywords. Community Mining, Community Detection, Graph Clustering, Spectral Clustering, Data Mining.

II. Introduction

Social network and social network data analysis are being pronounced more and more in today's literature in data mining, graph mining, machine learning, and data analysis. One of the prevalent problems is detecting communities and their overlaps. Communities or cluster are vertices in a graph with high degree of connectivity between them which stands them out from the rest of the graph. Some of the community detection algorithms have used notion of edge-betweenness for detection of communities as the density of edges between nodes that belong to a community is greater than density of edges between nodes that don't form a community. The main problem faced with community detection is time complexity of running conventional methods on giant modern social network graphs with billions of edges. Some of the researchers have done their clustering algorithms locally on the social graphs in order to reduce the complexity of their algorithms. Community detection algorithms are expected to be scalable considering the ever-growing social networks. Some of the important features[1] of the communities are considered to be as follows:

1) $Overlapping$: communities can overlap in which users share the same interests and have the same edges in common between two or more communities.
2) $Directed$: edges within a community can be directed or undirected. In terms of social networks, we can consider all of the edges to be directed.
3) $Weighted$: edges in the communities can be weighted to denote that various users have different affiliations and interaction rate with the community. The more influence a node brings up to a community, the greater is its edge weight connecting the node to the community.
4) $Multi-dimensional$: interactions within a community can be multi-dimensional, meaning that people can use various methods to interact with each other by posting, sharing, liking, commenting, tagging, etc.
5) $Incremental$: communities and community detection algorithms are expected to be incremental in which adding a new node and assigning a community to it, would just need a local search for the node in its neighborhood. We definitely dont want to run the whole algorithm from the beginning just for finding a community for a newcomer node.
6) $Dynamic$: communities can be dynamic and evolve through the time. Because most of giant social networks are dynamic, many of researchers have proposed the idea of streaming graph partitioning [15] which can be done using distributed computations and are mostly known as one-pass algorithms. In one-pass algorithms each node is assigned to a partition upon arrival in a greedy manner such that the objective function of the partition in graph is maximized.

Community detection approaches in the literature mostly rely solely on the link analysis and ignore the available information in the modern social networks. As an example, Twitter entails lots of metadata like user location, age, gender, geotags, interests all of which can be used in clustering.

III. Motivation

One of the benefits of community detection is refining the recommender systems by suggesting the products used by some members of the overlapping community to the rest. As an example to illustrate the importance of overlapping community detection consider the LinkedIn as your social network. In LinkedIn each person can be part of various communities. So now suppose two LinkedIn friends, A and B, that at least have one community in common C_{AB}, say University of Wisconsin-Madison. If A now works as a recruiter in Facebook, $C_{Facebook}$, theres a chance that B might be interested in a job offer from A to work in Facebook. This applies to person B or any of his friends in any overlapping community A is a member of. The reason is that people in overlapping communities, share the same attributes and we can predict that they will like each others' attribute with a large probability. As another example, consider all the frequently co-purchased items in Amazon. If we build a network of Amazon products in which an edge shows the product being co-purchased together in which communities can be verified by the departments each item belongs to and tags assigned by the users, sellers or the Website. We can predict what items should be recommended to the customer or customers with similar tastes in shopping which will enhance the shopping experience of users. In communities we find lots of triangle structures and one of the well-known methods for detecting communities is detecting the triangles within them. However due to the huge billion node structure of modern social network graphs, the adjacency matrix of their associated graphs are so giant that makes the conventional clustering methods based solely on vertices infeasible. The approach that we have adopted is clustering edges rather than solely nodes. This way we will also be able to recognize the overlapping communities. Due to the noise in the topology structure, it is also very wise to make use of node attributes in the network, to add some extra links to the network and predict the communities based on both edge clustering as well as node attribute similarities.

Event detection and event prediction are also two other important motivations for finding overlapping communities. Consider two Facebook groups Computer Engineering and Computer Science. For both of these groups people should have a valid @wisc.edu email to join which mean all the users belong to the community of university of Wisconsin-Madison. Considering the fact that students in both of the fields have a lot of interests in common, if there's an event announced by one of the members in Computer Science Facebook group which has some overlapping connections in Computer Engineering Facebook group, there's a great chance that this event would be of interest to the rest of Computer Engineering group members who even don't have a common connection with that person.

Spam and malicious user detection detection are two other important motivations for detecting communities and their overlaps. Once communities are detected, one can detect the nodes that have different behaviours than nodes inside the communities. In [21] authors have proposed the idea of enhancing sentiment analysis using community detection in Twitter. There are two main methods for semantic analysis: machine learning and semantic orientation. Using the former method a classification model should be trained for differentiating between different semantic classes while in the latter usually there exists a dictionary of subjectively meaningful words for scoring the documents subjective content. In this work they first cluster the tweets and detect the communities and afterwards use the SentiWordNet[22] lexicon for gauging the sentiments of each cluster. Before doing the sentiment analysis, authors did some preprocessing on the tweets as follows: converting all the characters to lowercase, removing hashtags, retweet indicators, URLs and punctuations, tokenizing the tweets into individual words for looking up in the aforementioned lexicon. While community detection and sentiment analysis are usually done separately in social network contexts, in this paper authors have integrated them and do the sentiment analysis in each individual community for achieving much accurate results.

IV. Related Work

There has been various approaches toward community detection that we have picked a few of them here for overview. Community detection problem falls into various categories of strategies. GN algorithm: GN, known as Girvan and Newman[16], hierarchically divides the graph to its partition using the edge betweenness metric. In GN links are iteratively removed from the graph based on the betweenness score and the algorithm continues till the modularity of resulting partition maximizes. Modularity achieved by GN algorithm is one of the good quality measurement techniques. A faster version of GN algorithm is starting with a null graph including only the nodes and adding the edges using greedy method such that maximizes possible modularity of GN algorithm in each step.

- $Infomap$: Communities can be detected based on random walks. The intuition behind random walk is the higher probability of remaining within the community through selecting an edge than going out of the community, as in proposed in Infomap[13], due to the fact that edge density is higher within the community. In Infomap network flow is modeled using the random walks in an undirected graph. A random walker is presumed to be walking inside a community if he is spending a considerable amount of time traversing among the nodes of the community. The directed version of Infomap is similar to Google's PageRank algorithm.
- LCA: Link cluster algorithm (LCA) is another category which deals with detecting communities based on edges rather than based on vertices. LCA considers the edges to be in the same community based on their similarities. LCA has the resolution problem in which links in the neighborhood of dense communities show vanishing similarities.
- SCD: Designed for detecting disjoint communities, in SCD notion of WCC is used which can be parallelized for speeding up the computations further. One of the community metrics is its WCC which is based on the fact that in large communities we have lots of triangles. SCD consists of two phases: first phase which creates a rudimentary clustering using clustering coefficients as a heuristic, the second phase includes moving the nodes around the communities for increasing the WCC of the communities. Probability of two nearest neighbors of a node, being nearest neighbors to each other, is considered as clustering coefficient. Such a behaviour forms triangles in a graph and usually within the communities we have high clustering coefficients depicting more triangles. One main problem with definition of clustering coefficient is that it depends on the global properties of the network. They claim that use of WCC and dealing with triangles rather than counting the edges matters in their speedup comparing to other state of the art algorithms. In order to

speed up their algorithm and consume less memory, they remove all the edges that are not part of a triangle. They sort the vertices decreasingly based on their clustering coefficients and when two vertices have the same clustering coefficient, they are sorted based on their degree. In an iterative fashion, in a set of vertices called partition P, they create a community for each vertex v and all of its not-already visited neighbors. Eventually they add the community C to the partition P. The last phase, partition refinement is done through the use of WCC concept which is based on hill climbing approach. They move vertices between the already partitioned communities until no further improvement in WCC of the partitions is achieved higher than a predefined threshold.

- $CESNA$: developed at Stanford's SNAP group is a community detection method which includes the node attributes along with clustering based on the graph topology. They literally build communities based on edge structure and node attributes (CESNA[17]). Using their model, one can also detect overlapping communities. Despite some of the literatures which assume that communities and node/edge attributes are marginally independent, they have supposed that communities create both network structure alongside with attributes which shows dependence between the network and attributes. One of the main reasons they have employed node attributes is compensating for missing data in presence of noise in the network structure.
- $BigCLAM$: In BigCLAM authors discuss the fact that despite many of the literature supposing there is less density in the overlapping communities that the rest of community without overlap, there is more density wherein communities overlap. According to the result in BigCLAM[7], community overlaps happen to be dense. They have studied their groundtruth networks and have realized that communities highly overlap which results in the fact that nodes in the overlap of two communities have more chances of being connected than in the rest of each community. In BigCLAM they show the node community membership using a bipartite affiliation network connecting nodes of the graph to the communities they belong to. Also they have incorporated the fact that people are affiliated with various communities differently into their algorithm and have given non-negative weights to each of the edges in the bipartite network. BigCLAM can be used for various types of network with non-overlapping, overlapping and nested communities.
- $CODICIL$: Another community detection tool which have used node attributes beside the network topology for overcoming the noise in the link structure is CODICIL. Noise in the links can be either incorrect links (false positive) or missing links (false negative). CODICIL creates content edges for improving the accuracy of the algorithm while decreasing the runtime and increasing its scalability. They create bigram of node attributes and based on the cosine similarity or Jaccard coefficient they find the nodes which are most similar and create content edges based on their similarity. In CODICIL they start with creating the content edges and continue with sampling the union of the topology edges and the content edges in a way to save only the edges that are locally in a neighborhood. Eventually they partition the simplified graph into communities.
- $Walktrap$[18]: Is based on a random walker trapped in a community and communicating with community member. The distance between two nodes is defined in terms of random walk process in which short random walks is taken between each pair of the communities. This algorithm finishes after $n - 1$ passes in which all of the nodes are combined into a single community. The output of this algorithm is maximum modularity network partitions. The intuition behind this algorithms is that if two nodes i and j are in the same community, in order to reach to node k, the distance shouldn't be very different using random walk. The walktrap clustering algorithm is implemented in the iGraph package[23].
- $Louvain$: Tries to optimize the modularity of network partitions using greedy optimization. The modularity is first refined locally in small communities, and then these smaller communities are considered as nodes and are aggregated into bigger communities iteratively until the maximum modularity is achieved.

V. Co-Clustering

Co-clustering, a powerful data mining approach for dyadic data, often known as biclustering identifies the structures and patterns in the latent class models that sparse data exists. This method is capable of detecting local patterns which might not be detected by other well-known unsupervised one-way clustering methods like K-means. In co-clustering rows of matrix M are clustered into X_r rows while the columns are clustered into X_c clusters simultaneously in a way that minimizes the sum-squared deviation from the mean inside each cluster. Co-clustering monotonically decreases the the loss in mutual information. In DI-SIM they have replaced SVD (singular value decomposition) for co-clustering the directed graphs with eigendecomposition used in symmetric undirected graphs. Authors' motivation for use of SVD is the emerge of big data and need for low rank approximation. SVD is a generalization of eigendecomposition and in case of having a symmetric squared matrix, SVD and eigendecomposition are equivalent. In DI-SIM they have used number of common parents and number of offspring as the similarity measure that correspond to $A^T \times A$ and $A \times A^T$ respectively in which A is the adjacency matrix of a directed graph. Spectral co-clustering clusters the word document matrix using the word document bipartite graph. They can produce high quality results but arent scalable. Most of the recent literature in this category try to sparsify the matrices and parallelize the procedure in order to make it more scalable. Co-clustering, an unsupervised learning method, which has the complexity of NP-hard, has been used widely in bioinformatics, text mining, market-based data analysis, natural language processing, recommendation system analysis, etc. In a social graph we might have noisy data in our node attributes or missing link in our network structure. Using co-clustering one can predict the missing values.

$DI - SIM$[19] co-clustering algorithm which can be generalized to any arbitrary matrix rather than just a very symmetric specialized matrix. Authors have talked about co-clustering in directed graphs which will help detecting the communities. Citation networks, Google+, Twitter, Friendfeed, etc are some well-known examples of asymmetric directed networks. Facebook is good depiction of undirected social graphs wherein all the relationships are symmetric. However, studying it deeply, even friendships in Facebook are implicitly asymmetric depending on who sends the friend request first and who accepts it afterward. Currently Facebook follow feature made it explicitly asymmetric directed graph. Apart from that, Facebook interactions are all considered to be asymmetric which makes clustering the directed social graphs more meaningful. Due to existence of extensive research on clustering the symmetric relations in graph, some of the researchers have aimed to symmetrize the matrices in order to use the known co-clustering algorithms.

VI. Spectral Clustering

Due to the fact that many of the clustering or community detection algorithms have stemmed from spectral clustering and it has played such an important role in the recent decades' literature, this section is dedicated to its details. Spectral clustering is one of the most frequently used traditional clustering methods, also known as hierarchical divisive clustering due to using a bottom-down approach for dividing nodes into clusters. i.e. initially all nodes are considered as one cluster and later are split into two and the process is repeated until reaching a threshold. Spectral clustering is performed on the laplacian matrix of the graph by calculating the eigenvectors and eigenvalues of it. Eigenvectors are calculated as

$$(D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}})x = \lambda x$$

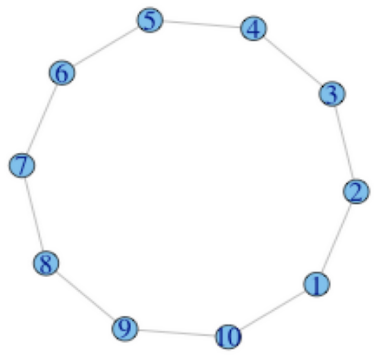
where $D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}$ is the Laplacian matrix. The sign of the elements of second eigenvector is used to split the nodes into clusters such that the value of normalized cut is least in the graph. The second eigenvector of the laplacian matrix is called Fiedler vector. In general Laplacian matrix gives similarity between the data elements in the matrix which processing assists in spliting the node into clusters. However for a giant graph, calculation complexity of eigenvector and eigenvalues is way more and has very high time complexity as it takes longer to converge. The traditional spectral clustering method is approximately infeasible for very large graph and the idea of distributed computation has been proposed to address this problem which is yet based on slow processing of random walks. Spectral clustering deals with connectivity instead of geometrical proximity. So if the data isn't very well geometrically-separated, but clusters aren't connected, spectral clustering will work well. Spectral clustering has been used in image, shape, color and motion segmentation, face recognition, image clustering, community detection and point correspondence. There are five methods of Laplacian calculations as following[20]:

- simple Laplacian is given by
  $$L = D - A$$
  where D is the degree matrix defined as
  $$d_i = \sum_{j=1}^{n} w_{i,j}$$
  in which only the diagonals have a value equal to weights of all the edges connected to the specific node and for an undirected matrix, D is symmetric. A is the affinity matrix an $N \times N$ matrix with elements as $A(i,j)$ depicting the affinity between the nodes i and j .
- normalized Laplacian is calculated as
  $$L_N = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$$
  which is used in the Ncut software developed by Shi and Malik used in our simulations.
- generalized Laplacian is calculated as
  $$L_G = D^{-1} L$$
- relaxed Laplacian
  $$L_\rho = L - \rho D$$
- Ng, Jordan and Weiss Laplacian in which $A_{i,i} = 0$ is calculated as
  $$L_{NJW} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

Stages of spectral clustering are as follows:

1) determine what needs to be clustered, like as in an image, decide on what feature to select (ex. points of interest, regions, or the whole image)
2) creating the similarity matrix as shown in an example in Figure 5.
3) create the Laplacian matrix L out of the graph as depicted in Figure 6.
4) finding the eigenvalues and eigenvectors of matrix L as calculated in Figure 7.
5) Map each point to a lower-dimensional representation based on one or more eigenvectors.

Advantages of Spectral clustering include no need for a distance measure, noise reduction, and that can be applied to most of real-life networks. On a contrary, its disadvantages are as follows: very slow, sensitive to image contrast, sensitivity to parameter choice like cluster number, neighborhood, similarity measure, etc., designed only for static data and not suitable for evolutionary data, difficulty in interpretation of clustering results, and being very computationally intensive for large datasets.

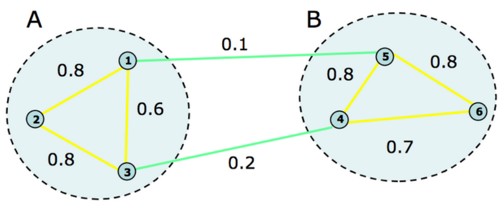


(a)

```
 [1,]  2 -1  .  .  .  .  .  .  . -1
 [2,] -1  2 -1  .  .  .  .  .  .  .
 [3,]  . -1  2 -1  .  .  .  .  .  .
 [4,]  .  . -1  2 -1  .  .  .  .  .
 [5,]  .  .  . -1  2 -1  .  .  .  .
 [6,]  .  .  .  . -1  2 -1  .  .  .
 [7,]  .  .  .  .  . -1  2 -1  .  .
 [8,]  .  .  .  .  .  . -1  2 -1  .
 [9,]  .  .  .  .  .  .  . -1  2 -1
[10,] -1  .  .  .  .  .  .  . -1  2
```

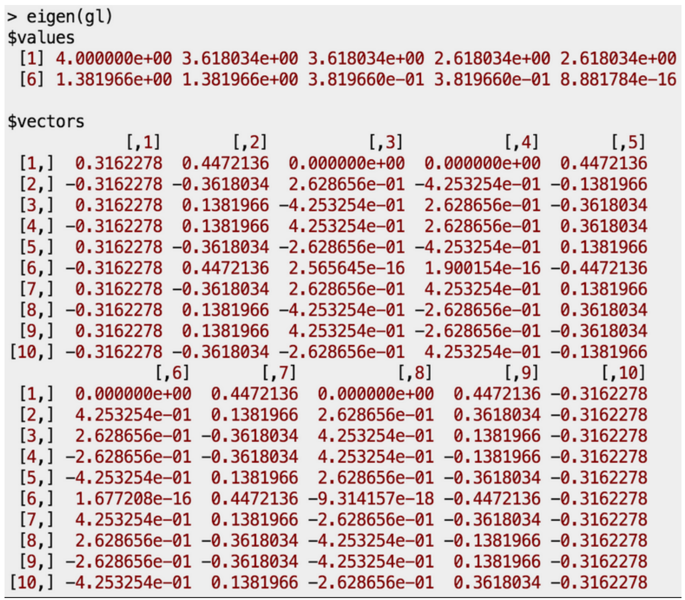
(b)

Fig. 2: (a) A ring graph (b) Its Laplacian matrix



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	0.8	0.6	0	0.1	0
x_2	0.8	0	0.8	0	0	0
x_3	0.6	0.8	0	0.2	0	0
x_4	0.8	0	0.2	0	0.8	0.7
x_5	0.1	0	0	0.8	0	0.8
x_6	0	0	0	0.7	0.8	0

Fig. 1: A weighted graph and its corresponding similarity matrix



```
> eigen(gl)
$values
 [1] 4.000000e+00 3.618034e+00 3.618034e+00 2.618034e+00 2.618034e+00
 [6] 1.381966e+00 1.381966e+00 3.819660e-01 3.819660e-01 8.881784e-16

$vectors
            [,1]       [,2]          [,3]          [,4]       [,5]
 [1,]  0.3162278  0.4472136  0.000000e+00  0.000000e+00  0.4472136
 [2,] -0.3162278 -0.3618034  2.628656e-01 -4.253254e-01 -0.1381966
 [3,]  0.3162278  0.1381966 -4.253254e-01  2.628656e-01 -0.3618034
 [4,] -0.3162278  0.1381966  4.253254e-01  2.628656e-01  0.3618034
 [5,]  0.3162278 -0.3618034 -2.628656e-01 -4.253254e-01  0.1381966
 [6,] -0.3162278  0.4472136  2.565645e-16  1.900154e-16 -0.4472136
 [7,]  0.3162278 -0.3618034  2.628656e-01  4.253254e-01  0.1381966
 [8,] -0.3162278  0.1381966 -4.253254e-01 -2.628656e-01  0.3618034
 [9,]  0.3162278  0.1381966  4.253254e-01 -2.628656e-01 -0.3618034
[10,] -0.3162278 -0.3618034 -2.628656e-01  4.253254e-01 -0.1381966
               [,6]       [,7]          [,8]       [,9]      [,10]
 [1,]  0.000000e+00  0.4472136  0.000000e+00  0.4472136 -0.3162278
 [2,]  4.253254e-01  0.1381966  2.628656e-01  0.3618034 -0.3162278
 [3,]  2.628656e-01 -0.3618034  4.253254e-01  0.1381966 -0.3162278
 [4,] -2.628656e-01 -0.3618034  4.253254e-01 -0.1381966 -0.3162278
 [5,] -4.253254e-01  0.1381966  2.628656e-01 -0.3618034 -0.3162278
 [6,]  1.677208e-16  0.4472136 -9.314157e-18 -0.4472136 -0.3162278
 [7,]  4.253254e-01  0.1381966 -2.628656e-01 -0.3618034 -0.3162278
 [8,]  2.628656e-01 -0.3618034 -4.253254e-01 -0.1381966 -0.3162278
 [9,] -2.628656e-01 -0.3618034 -4.253254e-01  0.1381966 -0.3162278
[10,] -4.253254e-01  0.1381966 -2.628656e-01  0.3618034 -0.3162278
```

Fig. 3: Eigenvalues and eigenvectors of the ring graph Laplacian matrix

VII. Community Detection Methods in R

R provides various packages for community mining. One of the most important ones, called iGraph[23], has various community detection algorithms implemented as follows:

- edge.betweenness.community: In this method edges are removed in the decreasing order of their edge betweenness score (number of the shortest paths going through each given edge). The intuition behind this algorithm is the fact that the edges between communities are having higher probability of being in multiple shortest paths as in many cases they are the only option to traverse between the different groups. This method has acceptable results but its computational complexity is very high because at every stage of the algorithm after the edge removal, the edge betweenness scores have to be re-calculated. Also another shortcoming of this method is that is yielding only the full dendrogram rather than specifying where to cut the dendrogram. In order to realize where to cut the dendrogram we have to use other metrics like modularity scores at each level of the dendrogram to obtain a better insight where to cut the dendrogram into communities.
  - Directed edges: TRUE
  - Weighted edges: TRUE
  - Runtime: $|V||E|^2$
- fast.greedy.community: In this bottom-up algorithm a quality function called modularity is being optimized in a greedy fashion. In the beginning, each single vertex depicts a community itself and iteratively communities merge in a way that each merge is locally optimal and results in largest modularity at each level. The stopping criterion of the algorithm is no further increase in the

modularity. Fast greedy algorithm not only gives the dendrogram but also provides the clustering. This method is really fast and is mostly used as a first approximation due to no need for tuning. The main disadvantage in using this method is the resolution problem in which the small communities get merged with their neighbor communities till reaching the stop criterion.

  - Directed edges: FALSE
  - Weighted edges: TRUE
  - Runtime: $|V||E|\log|V|$

- walktrap.community: This method is based on random walks and the intuition behind using it is if you randomly walk in the graph, it is very likely that you stay within the same cluster because according to the definitions, density of edges within a community is really high and the chance the an edge take you to another member of the same community is higher than an edge taking you outside of the community and taking random walks takes longer within a community. Using the result of the these random walks the algorithm merges separate communities in a bottom-up manner like that of fastgreedy.community algorithm. Modularity score can be used for choosing where to cut the dendrogram. Comparing to fastgreedy.community algorithm, randomwalk.community is slower but more accurate.
  - Directed edges: FALSE
  - Weighted edges: TRUE
  - Runtime: $|V||E|^2$
- spinglass.community is an approach from statistical physics, based on the so-called Potts model. In this model, each particle (i.e. vertex) can be in one of c spin states, and the interactions between the particles (i.e. the edges of the graph) specify which pairs of vertices would prefer to stay in the same spin state and which ones prefer to have different spin states. The model is then simulated for a given number of steps, and the spin states of the particles in the end define the communities. The consequences are as follows: 1) There will never be more than c communities in the end, although you can set c to as high as 200, which is likely to be enough for your purposes. 2) There may be less than c communities in the end as some of the spin states may become empty. 3) It is not guaranteed that nodes in completely remote (or disconencted) parts of the networks have different spin states. This is more likely to be a problem for disconnected graphs only, so I would not worry about that. The method is not particularly fast and not deterministic (because of the simulation itself), but has a tunable resolution parameter that determines the cluster sizes. A variant of the spinglass method can also take into account negative links (i.e. links whose endpoints prefer to be in different communities). leading.eigenvector.community: A top-down hierarchical method for optimizing the modularity function that in each step we split the graph into two subgroups such that the splits itself gains a decent increase in the modularity value. The split is decided by computing the leading eigenvectors of the modularity matrix. The stopping criterion prevents tightly connected clusters to be split more. If we have a degenerate graph, the algorithm might not work because of eigenvector computations if the ARPACK eigenvector is unstable. However if the graph is non-degenerate, there are high chance of achieving a greater modularity score than that of the fastgreedy.community method while being tad bit slower.
  - Directed edges: FALSE
  - Weighted edges: FALSE
  - Runtime: $c|V|^2 + |E|$
- label.propagation.community: In this algorithm each node is given a label out of the K available labels. This method iteratively re-assigns labels to the nodes such that every node receives the most frequent label of all its existing neighbors synchronously. If the label of each of the nodes is the same as the most frequent label in their neighborhoods. This algorithm is very fast but doesnt always converge to the same final result due to various initial random configurations. To overcome this shortcoming, one can run the algorithm a large number of times per graph for building consensus labeling but its a really tedious task.
  - Directed edges: FALSE
  - Weighted edges: TRUE
  - Runtime: $|V| + |E|$
- infomap.community: add here
  - Directed edges: TRUE
  - Weighted edges: TRUE
  - Weighted nodes: TRUE
  - Runtime: none given; looks worst case like $|V|(|V| + |E|)$ based on quick reading.

VIII. Evaluation Methods

For the purpose of evaluation we need networks with groundtruth communities. Yang and Leskovec [3] have created benchmarks with real datasets in which groundtruth communities are specified many of them have been used in the very recent literature as a method for evaluation. These datasets can be accessed in SNAP group at Stanford and are open to public. They have various kind of networks which serves different purposes in small, medium or very large size.

Quality functions can be used when there's no groundtruth for the communities to assess the quality of detected communities. Three of the most useful quality functions are as follows:

- *Coverage* is defined as ratio between the intra-community edges and all the edges in the graph and is one of the simplest measures for community quality which is biased towards coarse-grained communities. Modularity is defined as having more internal edges and less external edges as is defined as

- *Modularity* measures the strength of each partition by considering the degree distribution. One main problem with modularity approach is that it cannot detect well-defined small communities when the graphs are extremely large.
- *Conductance* is considered as ratio of number of edges within the community over those leaving it. Conductance is a good measure because problem of detecting communities can be seen as minimum cut problem in which a cut is a partition of vertices within a graph that can set the graph into two or more disjoint sets.

  Other important metrics used are *WCC* which is weighted community feature. They higher the WCC, the better is the quality of the detected community, *NMI* which is normalized mutual information accounts for the overlap between the communities, *Jaccard index*, *Cosine similarity*, and *F1 score* which all can be used to detect the quality of the community detection algorithms and their overlaps at node level.

IX. Future Work

As data can get so large, distributed data mining algorithms are becoming more pronounced. In $DisCo$ [11] they have used distributed co-clustering using Map-Reduce programming model. Authors have stored their adjacency list as a list of key-value pairs in the HDFS. They initialize two Map-Reduce jobs for iterating between rows and columns as well as the synchronization step for the purpose of updating the global parameters and run them on the hadoop cluster. As we have selected modern social networks as our target, due to huge number of edges, use of distributed co-clustering algorithms will be essential to the improvement of our research. Also detecting communities and overlapping communities in tripartite graphs which are hypergraphs consisting of users, resources and tags and each hyperedge (u,t,r) denotes that a user u has assigned tag t to the resource r has many applications in networks like last.fm and pandora.com. In such networks, each node usually belongs to multiple communities and detecting overlapping communities is of higher importance for recommending new resources and new friends to users. Also recently some community mining researchers have shifted the gears towards detecting evolution in the community and predicting the evolution in the future.

References

[1] M. Cosica, F. Gianootti, and D. Pedreschi,“ A Classification for Community Discovery Methods in Complex Networks,” *CoRR* as/1206.3552 (2012)

[2] F. Moradi, T. Olovsson, P. Tsigas, “ An Evaluation of Community Detection Algorithms on Large-Scale Email Traffic,” *SEA* 283-294 (2012)

[3] J. Yang, and J. Leskovec, “Defining and Evaluating Network Communities Based on Ground-Truth,” *ICDM* 745-754 (2012)

[4] Y. Song, and S. Bressan, “Fast Community Detection,” *DEXA* 404-418 (2013)

[5] A. Prat-Prez, D. Dominguez-Sal, J. M. Brunat, and J. Larriba-Pey, “ Shaping Communities out of Triangles ,” *CoRR* abs/1207.6269 (2012)

[6] A. Prat-Prez, D. Dominguez-Sal, and J. Larriba-Pey, “ High Quality, Scalable and Parallel Community Detection for Large Real Graphs ,” *WWW*, (2014)

[7] J. Yang, and J. Leskovec, “Overlapping community detection at scale: a nonnegative matrix factorization approach,” *WSDM* 587-596 (2013)

[8] Y. Ruan, D. Fuhry, and S. Parthasarathy,“ Efficient Community Detection in Large Networks using Content and Links,” *CoRR* abs/1212.0146 (2012)

[9] G. Csardi, and T. Nepusz, “The igraph software package for complex network research,” *IJCS* (2006)

[10] Stanford Network Analysis Project https://snap.stanford.edu/

[11] S. Papadimitriou, and J. Sun, “DisCo: Distributed Co-clustering with Map-Reduce: A Case Study towards Petabyte-Scale End-to-End Mining,” *ICDM* 512-521 (2008)

[12] INFOMAP code http://www.mapequation.org

[13] M. Rosvall, and C. T. Bergstrom, “Maps of random walks on complex networks reveal community structure,” *PNAS* 1118 1123 (2008)

[14] A. Chakraborty, S. Ghosh, and N. Ganguly, “ Detecting overlapping communities in folksonomies ,” *HT* 213-218 (2012)

[15] C. E. Tsourakakis, C. Gkantsidis, B.z. Radunovic, and M. Vojnovic, “Streaming Graph Partitioning for Massive Scale Graphs,” *Microsoft Research Technical Report* (2012)

[16] S. Fortunato, and A. Lancichinetti, “Community detection algorithms: a comparative analysis,” *VALUETOOLS* (2009)

[17] J. Yang, J. J. McAuley, and J. Leskovec, “Community Detection in Networks with Node Attributes,” *CoRR* abs/1401.7267 (2014)

[18] P. Pons, and M. Latapy, “Computing Communities in Large Networks Using Random Walks,” *ISCIS* 284-293 (2005)

[19] K. Rohe, and B. Yu, “Co-clustering for Directed Graphs; the Stochastic Co-Blockmodel and a Spectral Algorithm,” *Technical Report at UW Madison Statistics Department* (2012)

[20] U. Luxburg,“A tutorial on Spectral Clustering,” *Statistics and Computing* 17(4),395-416 (2007)

[21] W. Deitrick, B. Valyou, W. Jones, J. Timian, and W. Hu, “Enhancing Sentiment Analysis on Twitter Using Community Detection,” *CN*, Vol.5 No.3 (2013)

[22] http://sentiwordnet.isti.cnr.it/

[23] , G. Csardi, and T. Nepusz, “The igraph software package for complex network research,” InterJournal, *Complex Systems* 1695. 2006. http://igraph.org