ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 8(31), DOI: 10.17485/ijst/2015/v8i31/70569, November 2015 ISSN (Online) : 0974-5645*

Overlapping Community Detection in Temporal Networks

Anupama Angadi[1] and P. Suresh Varma[2]

[1]IT Department, GMR Institute of Technology, Rajam - 532127, Andhra Pradesh, India; anupamaangadi.a@gmail.com,
[2]Department of CSE, University College of Engineering, Adikavi Nannaya University Rajahmundry - 53329, Andhra Pradesh, India; vermaps@yahoo.com

Abstract

Background/Objectives: One of the most commonly observed features of Online Social Networks is Community Structure. This feature provides great benefit focusing on insights of network structure, hidden patterns and the flow of information between actors. **Methods/Statistical Analysis:** Most real-world social networks are inherently dynamic, grow rapidly in terms of social interactions. These interactions in network are reflected by edges in a graph. Instead of updating a network structure for every edge change, the proposed method tracks the edges at every unique time stamp in a subgraph and modify the network only with the changed edges. **Findings:** There are many static community detection algorithms for discovering communities in networks, but very few deal with incremental structural changes in the network. The proposed algorithm DOMLPA (Dynamic Overlapping Multi-Label Propogation) deals with dynamic networks where data arrives as a stream to find the overlapping nodes in communities. To find the new edges the proposed algorithm lists out the differences in edges between the subgraph and the network for every snapshot. Based on the differences the label edges would be added or removed from the network and adjacency entries, neighbors' list and label distribution entries are modified eventually. Speaker node function is activated to start the propogation process inorder to get the labels for every node. If a node contains a only one label it belongs to single community. If a node carries more than one label with more than one maximum probability entry, then it belongs to multiple communities. The opportunity to capture the evolutionary patterns in dynamic networks is lost by not considering the time of interaction in static algorithms. **Application/Improvements:** The study of communities is helpful in examining patterns leading to understand the structure of networks, finding the information flow and events taking place between a group of social actors over a period of time and to identify trending sentiments about brands based on tweets.

Keywords: Actors, Community Detection, Overlapping Nodes, Social Networks, Temporal Networks

1. Introduction

Networks are every where. Study their structures is an interesting task for a wide set of fields. A community is a set of nodes between which the interactions are frequent. People in social networks are considerably characterized by multiple community memberships. For illustration, a person usually has ties to several social groups like family, friends, and colleagues; a researcher may be participating in various specializations. Farther, in online social networks, the number of communities an individual can belong to is essentially unlimited because a individual can simultaneously link up with as many groups as he cares. This also happens in other complex networks such as biological networks, where a node might take part in multiple functions.

Most algorithms[2,3,6,7,14,16,17] are planned to detect disjoint communities. To identify communities in static networks the authors implemented various social concepts like Betweenness Metric[7], Betweenness and Modularity Metric[7], Label Propagation[2,17,18] and Markov Chains[14,16]. Restricting communities to disjoint may not yield beneficial outcomes since communities in online social networks may naturally overlap. For this

* *Author for correspondence*

cause, many overlapping community detection methods have been suggested and used several approaches like K-Clique[11], Dendogram[1], Modularity function as Fitness function[8], Bayesian Non-Negative Matrix Factorization[13] and Label Propagation[12].

Networks are not static objects, where the edges appear and disappear over a period of time. Community structures change from time to time in evolving networks. In such case, static community algorithms cannot find essential communities from dynamic networks and so, the necessity of dynamic community detection[3-5,10,18] arises. The purpose of this work is to find the overlapping nodes from communities in a dynamic social network.

The principal contributions of the paper are summarized as follows. First, the framework for revealing the overlapping nodes via Dynamic Overlapping Multiple Label Propagation. Second, we present the evolving network with an instance. Experimental results on dynamic data set shows how the framework can trap the temporal event changes and identify overlapping nodes.

2. Dynamic Overlapping Multi Label Propagation Algorithm

Our approach is based on the algorithm DMLPA[9] which finds the disjoint communities in dynamic networks, receives a single label from listener face the problem of random tie breaking when selecting a label from set of labels having the same degree. Due to this the propagation produces non-deterministic outputs. To avoid this, our proposed approach presents a multi-label propagation algorithm, that allows all labels for propagation. If a node contains only one label with maximum probability it belongs to one community. If more than one label with maximum probability exists for a node, it belongs to many communities and it is said to be overlapping. In our example in Figure 2 (c) nodes 3, 5, 8 have more than one label with maximum probabilities they are marked as overlapping nodes and the rest of the nodes have a single label in Table 1.

The modifications in network reflect in the edges alone. The process of tracking temporal events of DOMLPA is: first, treat the graph G_0 (Figure 1 (b)) as a network and run the algorithm MLPA, to get the communities for the first snapshot(Figure 1 (c)); second, for the remaining timestamps, (step 2 in Algorithm 1) get the edge list (Figure 2 (a)); third, for each edge in the edge list according to the label, edge would be added or deleted from the sub graph G_t (step 4 to step 6 in Algorithm 1); fourth, find the difference between sub graph and network to know the remainder edges (Figure 2 (b)), (step 7 in Algorithm 1); finally we update the network only with the remainder edges (Figure 2 (c)).

The process of updating is: first, for every remainder edge in the edge list get the vertex and verify it is in list of vertices of the network, if it is, add that edge to the network and set the adjacency entry to 1, update the neighbors' list and probability distribution entry, (step 1 to 4 in Algorithm 2); Second, if the vertex is not in the list of vertices of the network add vertex to the network and do the remaining process similar as in (step 1 to 4 in Algorithm 2); third, if the label is delete remove the edge from the network and set adjacency entry and probability distribution entry to 0, (step 8 to 9 in Algorithm 2); finally call the speaker function to find the overlapping nodes.

2.1 Algorithm 1: DOMLPA

Input: Graph Timestamps(0…..T)

- run MLPA for G_0
- for t in 1 thru T:
- E = edgelist
- for each e in E
  - if e is an addition
    - add e to G_t
  - else
    - remove e from G_t
  - end if
- end for
- get the subgraph G_t
- ΔE = graph.difference(G_t ,G_{t-1})
- updateNetwork(G_{t-1}, E)
- end for

2.2 Algorithm 2: Update Network (G,E)

Input: Graph G, edge changes E

- for each e in E
- u,v= nodes of e
- x=u
- if x is in list of edges of G and label="add"
  - add e to G
  - A_e<-1
  - Update neighbor list of x
  - K<- sum of all adjacent neighbors of x
  - P_e<-1/K

Speaker(x)

- else:
- if x is not in list of edges of G and label="add"

addVertex x to G
$A_{(x,x)}$<-1
add e to G
add x to A
A_e<-1
Update neighbour list of x
K<- sum of all adjacent neighbors of x
P_e<-1/K
Speaker(x)

- else:
- if label="delete"

delete e from G
A_e<-0
P_e<-0
Speaker(x)

- end if
- end for

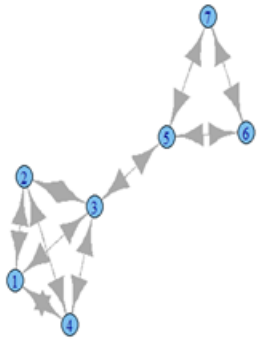
2.3 Algorithm 3: MLPA(G,E)

Input: Network N = (V, E)
Output: Community structure C = {C1, C2…CP}
T: user defined maximum iteration
r: threshold for post processing
q: threshold for conditional update
in: inflation operator

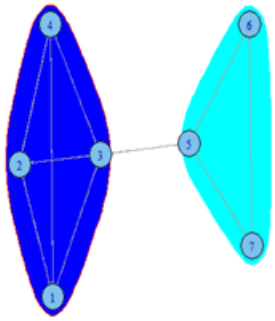
1. Give initial unique label names for each node (same as node id)
2. Add self loop to all the nodes in the network Sort nodes in the order of the degree and store it in ordered vector Calculate initial probabilities such that equal probabilities are assigned for each neighbour and store it in label distribution matrix
3. Repeat T times or until stop criterion satisfied All nodes in ordered vector are marked as unvisited repeat
   - Select a unvisited node
   - Each neighbour of the selected node sends a label or multiple labels based on the probability of occurrence (frequency) of labels in the memory based on speaking rule
   - Listener accepts one or more labels based on listening rule for updating process
   - If the update condition is satisfied, listener updates the probability of one or more labels and the listener is marked as visited. Inflation operator is applied to increase the probabilities of labels having maximum occurrence until all nodes in ordered vector are marked as visited
4. Post-process the labels in the memory based on the threshold r and form the communities. If a node contains multiple labels, overlapping communities occur.

```
      [,1] [,2]
 [1,] "1"  "2"
 [2,] "1"  "3"
 [3,] "2"  "1"
 [4,] "2"  "3"
 [5,] "3"  "1"
 [6,] "3"  "2"
 [7,] "3"  "4"
 [8,] "4"  "3"
 [9,] "4"  "5"
[10,] "4"  "6"
[11,] "4"  "7"
[12,] "5"  "4"
[13,] "5"  "6"
[14,] "5"  "7"
[15,] "6"  "4"
[16,] "6"  "5"
[17,] "6"  "7"
[18,] "7"  "4"
[19,] "7"  "5"
[20,] "7"  "6"
```

(a)



(b)



(c)

Figure 1. The evolutionary network from time to time. (a) The edge list for the first time stamp. (b) The corresponding snap shot of network G_0. (c) Communities identified by the MLPA algorithm on first snapshot.

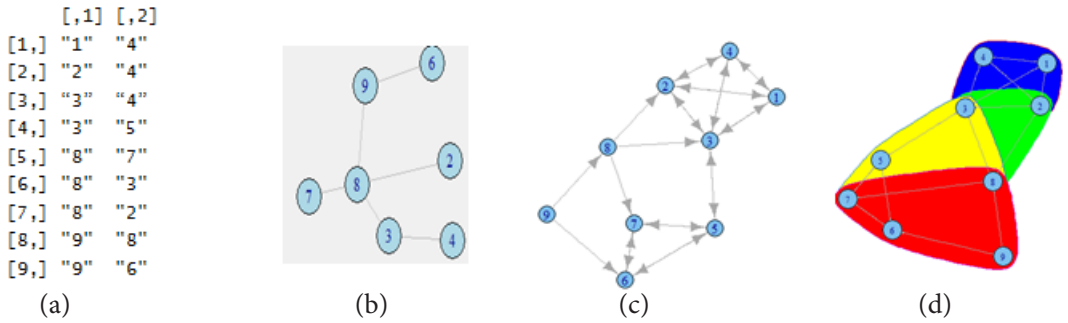


Figure 2. (a) The edge list for the next time stamp. (b) The graph difference between original graph G_0 and the subgraph contains edge list. (c) Updated Network with remainder edges. (d) Communities identified by the MLPA algorithm on second snapshot where colors represent communities

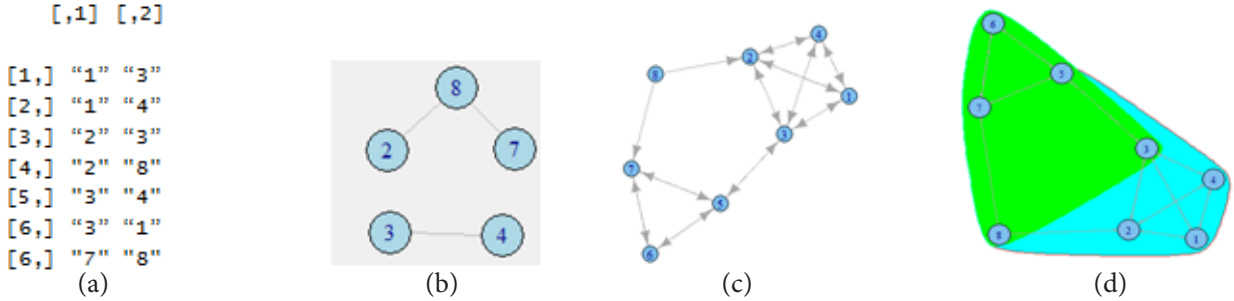


Figure 3. (a) The edge list of third time stamp. (b) The graph difference between G_1 and subgraph with edge list. (c) The final graph after updation, some edges were deleted. (d) Identified communities in G_2.

Table 1. The label distribution of each node in the network G_2. The number of labels reduces due to their low probabilities

Label	Iteration #	Iteration #
1	-	-
2	-	-
3	1,2,4	1,2,4
4	-	-
5	5,6,7	5,6,7
6	-	-
7	-	-
8	1,4,6	1,3,4,5,6

3. Evaluation and Results

The proposed framework is applied on Activities dataset available in MIT Human Dynamics Lab Dataset[19]. The data shows, on which user id and a campus the survey was conducted. The information in the dataset made it possible to create evolutionary network to identify overlapping nodes in the communities. Our timestamps for this network are months, that show in which month survey was conducted, on how many users, and on which campuses. Almost every userid is repeated in campus so, we found many overlapping nodes participated in both communities.

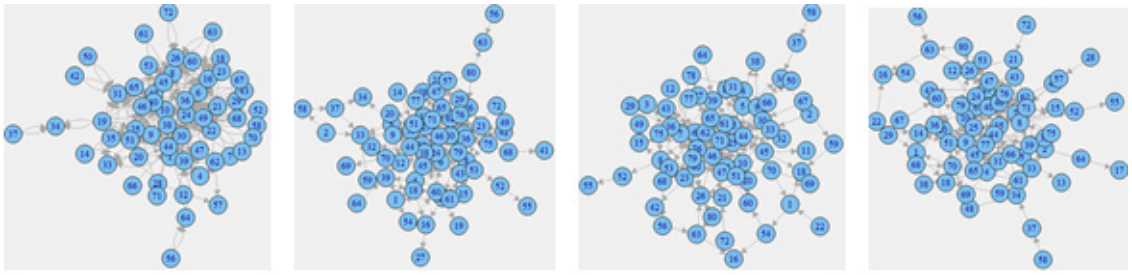


Figure 4. The sub graphs of every time stamp in evolutionary network for Table 2.

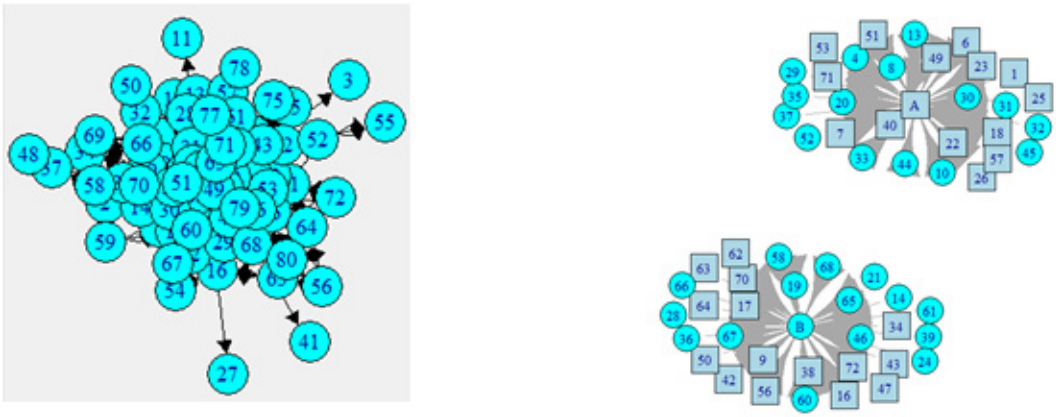


Figure 5. Community result shows the whole network partitioned into two communities. Here the shape square represenst the overlapping node and the circle represents the disjoint node.

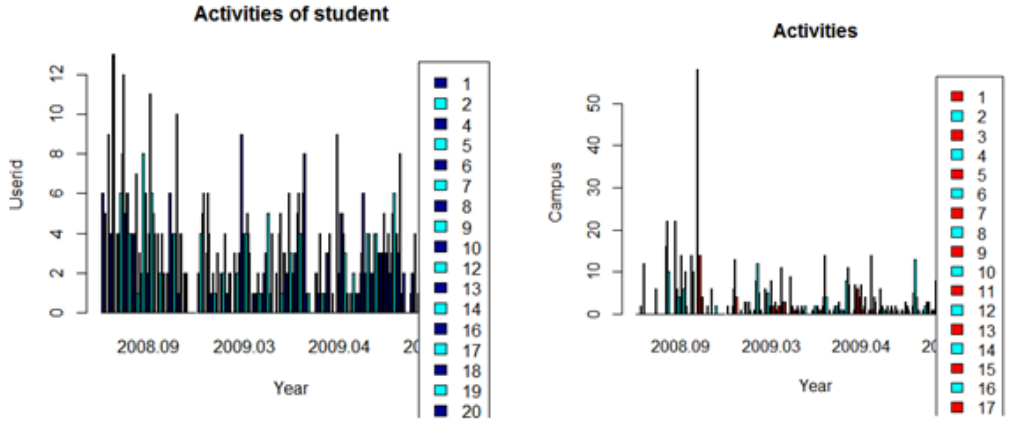


Figure 6. The evolutionary events on activities data set.

Table 2. The Time stamps in activities data set

Survey month	Vertices	Edges
September 2008	59	248
March 2009	68	145
April 2009	71	150
June 2009	70	152

4. Conclusion and Future Works

With the rapidly expanding social networks, it is challenging to analyse communities using dynamic algorithms. We have shown that our framework detects the overlapping community structures in evolving networks. We have shown through our experiments on the Activities dynamic data set, the number of changes over a period of time is a deciding factor in a runtime environment. With this, it is advisable to run DOMLPA to track changes occurred in the network and to keep the community structure up to date. Based on neighbors links every node is forced to belong to a community, a node with strong and more links become a leader and a node with weak links become an outlier. In our future work, we plan to extend our framework to evaluate the role and importance of nodes in a community.

5. References

1. Ahn YY, Bagrow JP, Lehmann S. Link communities reveal multiscale complexity in networks. Nature. 2010; 466:761-4.
2. Angadi A, Verma PS. Community detection in temporal networks. IJETCAS. 2015: 281-5.
3. Aston N, Hu W. Community detection in dynamic social networks. Communications and Network. 2014; 6: 124-36.

4. Aston N, Hertzler J, Hu W. Overlapping community detection in dynamic networks. Journal of Software Engineering and Applications. 2014; 7: 872-82.
5. Bhat SY, Abulaish M. HOCTracker: Tracking the evolution of hierarchical and overlapping communities in dynamic social networks. IEEE Transactions on Knowledge and Data Engineering. 2014.
6. Fortunato S. Community detection in graphs. Physics Reports. 2010; 75-174.
7. Girvan M, Newman MEJ. Community structure in social and biological networks. Proceedings of National national Academy of Science; USA. 2002; p. 7821-6.
8. Lancichinetti A, Fortunato S, Kertesz J. Detecting the overlapping and hierarchical community structure of complex networks. New J Phys. 2009.
9. Lee C, Reid F, McDaid A, Hurley N. Detecting highly overlapping community structure by greedy clique expansion. Poster at KDD. 2010.
10. Nguyen NP, Dinh TN, Shen Y, Thai MT. Dynamic social community detection and its applications. PLoS ONE. 2014.
11. Palla G, Derenyi I, Farkas I, Vicsek T. Uncovering the overlapping community structure of complex networks in nature and society. Nature. 2005; 435:814–8.
12. Prabavathi GT, Thiagarasu V. Design and development of overlapping community detection algorithm using Multi-Label Propagation. International Journal of Advance Research in Computer Science and Management Studies. 2014; 195-9.
13. Psorakis I, Roberts S, Ebden M, Sheldon B. Detecting overlapping communities in networks via non-negative factorization. Phys Rev. 2011.
14. Satuluri V, Parthasarathy S. Label graph clustering using stochastic flows: Applications to community discovery. SIGKDD. 2009; 737-46.
15. Sun H-L, Haung J-B, Tian Y-Q, Song Q-B, Liu H-L. Detecting overlapping communities in networks via dominant label propagation. Chin Phys. 2015.
16. Van Dongen S. A cluster algorithm for graphs. National Research Institute for Mathematics and Computer Science. 2000.
17. Xie J, Szymanski BK. Labelrank: A stabilized label propagation algorithm for community detection in networks. IEEE Network Science Workshop; West Point, NY. 2013. p. 138–43.
18. Xie J, Chen M, Szymanski BK. LabelrankT: Incremental community detection in dynamic networks via label propagation. 2013.
19. Social Evaluation. 2015 Apr 27. Available from: http://realitycommons.media.mit.edu/socialevolution1.html.