9-2016

Efficient community maintenance for dynamic social networks

Hongchao QIN
Northeastern University

Ye YUAN
Northeastern University

Feida ZHU
Singapore Management University, fdzhu@smu.edu.sg

Guoren WANG
Northeastern University

DOI: https://doi.org/10.1007/978-3-319-45817-5_50

Published in Web technologies and applications: 18th Asia-Pacific Web Conference, APWeb 2016, Suzhou, China, September 23-25: Proceedings. pp. 478-482. https://doi.org/10.1007/978-3-319-45817-5_50

Efficient Community Maintenance for Dynamic Social Networks

Hongchao Qin[1,2(✉)], Ye Yuan[2], Feida Zhu[1], and Guoren Wang[2]

[1] School of Information Systems, Singapore Management University, Singapore, Singapore
[2] School of Computer Science and Engineering, Northeastern University, Shenyang, China
qhc.neu@gmail.com

Abstract. Community detection plays an important role in a wide range of research topics for social networks. The highly dynamic nature of social platforms, and accordingly the constant updates to the underlying network, all present a serious challenge for efficient maintenance of the identified communities—How to avoid computing from scratch the whole community detection result in face of every update, which constitutes small changes more often than not. To solve this problem, we propose a novel and efficient algorithm to maintain the communities in dynamic social networks by identifying and updating only those vertices whose community memberships are affected. The complexity of our algorithm is independent of the graph size. Experiments across varied datasets demonstrate the superiority of our proposed algorithm in terms of time efficiency and accuracy.

Keywords: Community detection · Dynamic · Heuristic · Modularity

1 Introduction

Communities are groups of network nodes, within which the links connecting nodes are dense but between which they are sparse. In many applications, the structure of networks evolves over time, so do the communities in them. Most existing works [1,3] identify communities in a static network. They define an objective function Q which measures the quality of the communities. As the number of all possible partitions of nodes [5] is exponential of the network size,

This research is partially funded by the National Research Foundation, Prime Ministers Office, Singapore under its International Research Centres in Singapore Funding Initiative and Pinnacle Lab for Analytics at Singapore Management University, National Natural Science Foundation of China (No. 61332006, 61332014, 61328202, 61572119, U1401256) and the Fundamental Research Funds for the Central Universities of China (No. N150402005, N130504006).

it is NP-hard to obtain the communities with the maximum Q. Heuristic algorithms, such as [2], are therefore usually proposed as a solution. Yet those algorithms typically need to recompute the entire result for each network change, even through the change affects only a small number of vertices, which renders the detection computationally unaffordable for graphs that are large and evolve fast.

We propose a novel algorithm to maintain communities in dynamic social network by identifying and updating only those communities affected in graph evolution. The complexity of our algorithm is independent of the graph size. We conduct extensive experiments on various datasets and demonstrate the efficiency and accuracy of our approach.

Consider an undirected graph $G = (V, E)$. Let $n = |V|$ and $m = |E|$ be the number of vertexes and the number of edges in G. The most general method of detecting communities is to maximize the quality function $\mathcal{Q}$ [3].

Definition 1 (Modularity). *Given a social network G and the partition $\mathcal{C}_G = \{C_G^1, C_G^2 ... C_G^k\}$. The **modularity** is a quality function of the community structure, it can be denoted by $\mathcal{Q}$ and*

$$\mathcal{Q}(\mathcal{C}_G) = \sum_{i=1}^{k} \left(\frac{l(C_G^i)}{m} - \left(\frac{d(C_G^i)}{2m}\right)^2\right) \tag{1}$$

where $l(C_G^i)$ is the total number of edges joining vertexes inside community C_G^i and $d(C_G^i)$ is the sum of the degrees of all the vertexes $\{V_i \| V_i = C_G^i \bigcap V\}$ in G. And $Q(C_G^k)$ denotes single modularity of C_G^k.

Definition 2 (Dynamic Social Network). *A dynamic network $\mathcal{G}$ is represented by a series of time dependent network snapshots*

$$\mathcal{G} = \{G_0, G_1, G_2, ...G_t, G_{t+1}, ...\}$$

where $G_{(t)} = (V_t, E_t)$ is the snapshot of the network at the time point t. The differences between two consecutive snapshots G_{t+1} and G_t is denoted by ΔG_t, which contain nodes' insertions(deletions) and edges' insertions(deletions).

Problem Definition. The goal of the community maintenance problem is to update the communities of all the vertexes in G when the graph G is changed by inserting or deleting edges (all the nodes' insertions or deletions can transform into edges' insertions or deletions). It means that given the dynamic social network $\mathcal{G} = \{G_0, G_1, ...G_t, G_{t+1}, ...\}$ and the communities at t time $\mathcal{C}_{G_t}$, the community structure at time point $t + 1$ is detected based on the community structure at the time point t and the changes ΔG_t. And ΔG_t is split into a sequence of only one edge's insertion or deletion so it is easy to consider.

2 The Proposed Algorithm

Algorithm for Edge Insertion: There are three kinds of edge insertion.

When one vertex of the edge is a new vertex, we can simply join this node to the community of the other side's. There may have new edges containing the new node, but we can match the later insertion with the following sub-algorithms.

When both sides of the new edge are in one community, if we don't change the community partition, the modularity will increase. It can be proved that we can not divide the community to have higher modularity.

When the two vertexes of the edge are in different communities, after the insertion, the communities may not change, or the two vertexes will be added to a new community. We can mark the most possible common community as $C_{common}(u, v)$, and it has the largest number of same neighbours for vertex u and v, i.e., $C_{common}(u, v) = \{C^i_{G_t} \in \mathcal{C}_{G_t} \| argmax(|N(u) \cap C^i_{G_t}| + |N(v) \cap C^i_{G_t}|)\}$.

After the insertion of edge (u,v), we can join both u and v in $C_{common}(u, v)$ and the modularity will increase if there holds a inequation and the complexity checking it is independent with the graph size. If the modularity increase after joining both u and v in $C_{common}(u, v)$ (marked as C_c), then we have

$$Q(C_c \cup u \cup v) + Q(C_{G_t}(u) \backslash u) + Q(C_{G_t}(v) \backslash v) > Q(C_c) + Q(C_{G_t}(u)) + Q(C_{G_t}(v)) \quad (2)$$

Considering the definition of Q, formula 2 can be transformed to an simple inequation and the complexity checking it is independent with the graph size.

If the inequations in formula 2 do not hold, the modularity will not increase whatever community the vertex u and v join in.

Algorithm for Edge Deletion: There are two kinds of edge deletion.

When the two vertexes of the edge are in different communities, we can simply remain the communities unchanged. The number of the inside edges of $C_{G_t}(u)$ and $C_{G_t}(v)$ will not change after the deletion of edge (u, v), and the sum degree of all the vertexes in $C_{G_t}(u)$ and $C_{G_t}(v)$ will increase, so the modularity will increase if the communities don't change.

When vetex u, v are in the same community C^*, after the deletion of edge (u,v), we split the community into two parts $\{C^*_1, C^*_2\}$ and the modularity will increase if the number of edges between the two partions is not larger than one equation. Suppose that the number of edges between the two partions is $d(C^*_1, C^*_2)$. If we need to split the community, it's obvious that u, v are in different community of $\{C^*_1, C^*_2\}$ and there holds $Q(C^*_1) + Q(C^*_2) > Q(C^*)$. Then we can take

$$d(C^*_1, C^*_2) > \frac{d(C^*_1)d(C^*_2) - d(C^*_1) - d(C^*_2) + 1}{2(m-1)} - 1 \quad (3)$$

The formula 3 is easy to calculate. If it holds, we can split the community C^* into two parts, otherwise we need not to change it.

3 Experiments

In the experiments, we collect four real world datasets named Blogs (6,803 edges), ego-Facebook (88,234 edges), soc-Epinions1 (508,837 edges) and com-DBLP

(1,049,866 edges). We get all the datasets from [4] and conduct the experiments on a server with Intel Core i7 @3.6 GHz CPU and 16 GB RAM.

We can use the FNM (Fast Newman) algorithm [5] to get the communities of G_0 in the static network. And we compare our method, marked as CM (Community Maintenance), with FNM and the QCA (Quick Community Adaptive) [6,7] algorithm in dynamic network. To maintain the communities in the network, we need to emulate changes to the graph. Suppose that from G_t to G_{t+1}, five percent numbers of the nodes change. So we randomly insert or delete nodes to modify the graph as large as 5 % between each point.

Accuracy: We investigate the accuracy of the algorithm. As the FNM algorithm compute completely in every step, the modularity of the graph is a little much higher than the other algorithms. But they can only get a approximation value of the modularity and the difference is small in both algorithms. From the result, we can know that our algorithm have good accuracy.

Efficiency: In terms of efficiency, we report the time cost of maintaining communities from G_t to G_{t+1}. Figure 1 shows the running time of the algorithms. We can see that the speed of our method CM is faster than the QCA algorithm. But the cost of FNM is not acceptable if the original graph is large.

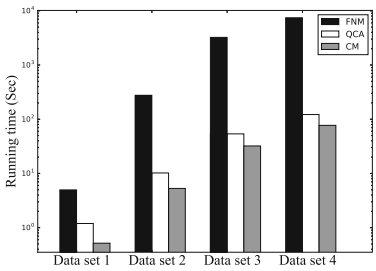

Fig. 1. Running time

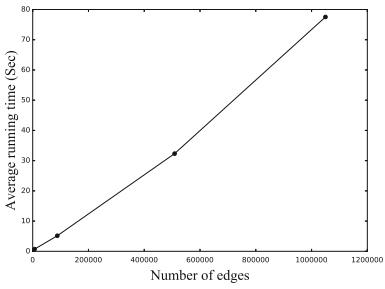

Fig. 2. Scalability of our algorithm CM

Scalability: To show the scalability of our algorithm, we average the running time of updating the communities from G_t to G_{t+1} with t ranges from 1 to 10 in different datasets. From Fig. 2, we can see that our algorithm CM perform well with the size of the dataset becomes larger.

4 Conclusion

We propose an efficient algorithm for maintaining the communities in dynamic social networks. We split the changes of the graph into edge insertions or deletions and discuss how they influence the communities. Extensive experiments demonstrate the efficiency and accuracy of our algorithm.

References

1. Blondel, V.D., Guillaume, J.L.: Fast unfolding of communities in large networks. J. Stat. Mech. Theor. Exp. **2008**(10), P10008 (2008)
2. Brandes, U., Delling, D., Gaertler, M.: On modularity clustering. IEEE Trans. Knowl. Data Eng. **20**(2), 172–188 (2008)
3. Fortunato, S., Castellano, C.: Community structure in graphs. In: Meyers, R.A. (ed.) Computational Complexity, pp. 490–512. Springer, New York (2012)
4. Leskovec, J., Krevl, A.: SNAP datasets: Stanford large network dataset collection, June 2015. http://snap.stanford.edu/data
5. Newman, M.E.: Fast algorithm for detecting community structure in networks. Phys. Rev. E **69**(6), 066133 (2004)
6. Nguyen, N.P., Dinh, T.N., Shen, Y., Thai, M.T.: Dynamic social community detection and its applications. PloS one **9**(4), e91431 (2014)
7. Nguyen, N.P., Dinh, T.N., Xuan, Y., Thai, M.T.: Adaptive algorithms for detecting community structure in dynamic social networks. In: 2011 Proceedings of IEEE INFOCOM, pp. 2282–2290. IEEE (2011)