Discovering Organizational Structure in Dynamic Social Network

Jiangtao Qiu  
School of Information  
Southwestern University of Finance and Economics  
Chengdu, China  
Jiangtaoqiu@google.com

Zhangxi Lin  
The Rawls College of Business Administration  
Texas Tech University  
Lubbock, TX  
zhangxi.lin@ttu.edu

Changjie Tang  
School of Computer Science  
Sichuan University  
Chengdu, China  
chjtang@vip.sina.com

Shaojie Qiao  
School of Information Science and Technology  
Southwest Jiaotong University  
Chengdu, China  
quashaojie@gmail.com

Abstract—Applying the concept of organizational structure to social network analysis may well represent the power of members and the scope of their power in a social network. In this paper, we propose a data structure, called Community Tree, to represent the organizational structure in the social network. We combine the PageRank algorithm and random walks on graph to derive the community tree from the social network. In the real world, a social network is constantly changing. Hence, the organizational structure in the social network is also constantly changing. In order to present the organizational structure in a dynamic social network, we propose a tree learning algorithm to derive an evolving community tree. The evolving community tree enables a smooth transition between the two community trees and well represents the evolution of organizational structure in the dynamic social network. Experiments conducted on real data show our methods are effective at discovering the organizational structure and representing the evolution of organizational structure in a dynamic social network.

Keywords—Organizational structure; Dynamical social network;

I. INTRODUCTION

Community discovery is a classical problem in social network analysis, where the goal is to discover related groups of members such that intra-community associations are denser than the associations between communities. Generally, a community is a sub-graph or a collection of members in a social network. It often reveals interesting properties shared by the members, such as common hobbies, social functions, etc. But Community discovery cannot reveal the power of members and the scope of their power in a social network. We find that organizational structure, a graph for illustrating the power of members and the scope of their power, is a versatile approach to solving the problem, for example, finding a leader from a terrorist network, exploring the spread of influence through a social network on viral marketing, and so on. However, to our best knowledge, none of existing works has ever addressed the organizational structure in dynamic social networks.

Methods on static community discovery mainly include hierarchical clustering [1,2,3], graph partitioning [4,5,6,7] and others [8,9]. Hierarchical clustering methods are aimed at finding natural divisions of a social network into groups, based on various metrics of similarity or strength of connection between vertices.

Many works exploited graph partitioning to find communities from a social network. In the simplest MINcut [4] algorithm, a connected graph (network) is partitioned into two sub-graphs, i.e. discovered communities, with the cut size minimized. However, MINcut often results in a very small sub-graph cut away. Various improved algorithm are proposed, such as normalized cut [5], ratio cut [6], min-max cut [7], etc. These methods only cut a graph into two parts. Typically a top-down technique is employed to recursively cut a graph to $k$ sub-graphs.

Many works examined the change of communities in a dynamic social network, called dynamic community discovery. However, they could not answer the questions related to the organizational structure, such as who is becoming more powerful, what the change of members’ power scope is, and so on. We believe that discovering the organizational structure from a dynamic social network is interesting and important. It may help, for instance, observing power change of members, and analyzing the transition of the relations among members. In this paper, we address the second issue on how to discover organizational structure in dynamic social networks.

The main contributions of this paper include:

(1) Proposing the community tree approach to the representation of organizational structure in a social network, which becomes solvable by combining the PageRank algorithm and random walks on graph.

(2) Conceiving a tree learning algorithm by employing the tree edit distance as a score function to generate the evolving community tree, which can well represent a smooth transition of organizational structure in a dynamic social network.

II. RELATED WORKS

Typically, sub-graphs.
Zhou et al [8] proposed two generative Bayesian models, combining probabilistic modeling with community detection, for semantic community discovery in a social network. Newman & Girvan [9] conceived an algorithm for discovering community structure from the networks with only a single type of vertexes and a single type of undirected and unweighted edges.

In analyzing a dynamic social network, most of studies [10,11,12] tried to obtain a snapshot of a single point in time in a dynamic social network or aggregate changes of a dynamic social network over a time window, then find static communities from the snapshot or the aggregation. However, both approaches may miss some important tendencies of dynamic networks.

Li et al [13] developed a hierarchical community model and Aaron et al [16] presented a general technique for inferring the hierarchical structure from a network. Different from the organizational structure proposed in this paper, hierarchical community model and hierarchical structure cannot reveal important information about organization in a network, for example, which immediate leader of a member is, or who supervises whom.

III. DISCOVERING THE ORGANIZATIONAL STRUCTURE IN THE STATIC SOCIAL NETWORK

Before analyzing the organizational structure of a social network, we loosely define a number of basic concepts. Organizational structure represents the power of each member and the scope of his power in a social network. If we can find the immediate leader of each member in a network and draw the leadership into a graph, the graph is then referred to as an organizational structure. We firstly provide a score, called m-Score, for each member, to represent the importance of the member in a social network. The higher the score is, the more important the member is. A data structure, namely community tree, is used to represent the organizational structure in a social network.

**Definition 1 (Community Tree):** Let \( P = \{p_1, ..., p_k\} \) be a collection of members, NULL denote the root of a tree, and each member in \( P \) be referred to as a node in the tree. For any member \( p_i \) in \( P \), it has a unique parent node \( p_j \) where \( m - \text{Score}(p_j) \geq m - \text{Score}(p_i) \). If the parent node of \( p_i \) is the root node NULL, \( p_i \) is called a core of the tree. A core and its descendents compose a community.

After derived the community tree from a social network, we can both discover communities and obtain organizational structure in the social network. An example of community tree is illustrated in Figure 1 where node 1 and node 5 are cores of community 1 and community 2 respectively, and node 1 is immediate leader of node 2.

PageRank may produce a global “importance” ranking for each web page by analyzing links between web pages. In this paper, we employ PageRank to evaluate “importance” of each member in a social network, which calculate m-Score for every node. In a network, random walks implicitly perform a soft clustering of nodes [14]. Therefore, we exploit random walks to find an immediate leader for each member. Associating random walks with m-Score of every node, a community tree can be derived from the social network.

A. Calculating m-Score

An improved PageRank algorithm for calculating the m-Score of members in a social network is described in Algorithm 1.

**Algorithm 1** calculating m-Score for members: calcScore
Input: social network \( G \)
Output: vector of m-Score for all members \( R \)
Steps:
1. \( R_0 \leftarrow S \)
2. Loop:
3. For each node \( p_j \):
4. \[ R_{k+1}[i] = \frac{1-d}{N} + d \cdot \sum_{p_j \in M(p_i)} W(p_{ij}) \cdot \frac{R_k[p_j]}{L(p_j)} \]
5. End
6. \( e \leftarrow ||R_k||_1 - ||R_{k+1}||_1 \)
7. \( R_{k+1} \leftarrow R_{k+1} + e \cdot S \)
8. \( \delta \leftarrow ||R_{k+1} - R_k||_1 \)
9. \( R_{k+1} \leftarrow R_k \)
10. While(\( \delta > \varepsilon \))
11. Normalize(\( R_{k+1} \))
12. Return(\( R_{k+1} \))

In calcScore, \( S \) is a start vector. Vector \( R_k \) stores m-Score of each member calculated in the \( k \)th iteration. \( R_k[i] \) is the m-Score of member \( p_i \), \( d \) denotes damping factor, and \( N \) denotes the number of members. \( M(p_i) \) denotes the collection where members are associated with \( p_i \). \( L(p_j) \) denotes the sum of weights for all edges associated with \( p_j \). \( W(p_{ij}) \) is the weight of the edge link \( p_i \) and \( p_j \). Let \( ||R_k||_1 \) be the \( L_1 \) norm of \( R_k \), we can calculate the error of \( \|R_k\|_1 \) and \( \|R_{k+1}\|_1 \), denoted as \( e \), and then calculate \( R_{k+1} \) again using \( e \). The iteration terminates when the \( L_1 \) norm of \( R_{k+1} - R_k \) is less than a preset threshold \( \varepsilon \). After \( R_{k+1} \) is normalized, the value of each element in vector \( R_{k+1} \) indicates the final m-Score of every member in the social network.
B. Deriving Community Tree

t-step random walks on graph sum the probabilities of all paths of length $t$ between two nodes. It gives a measure of the volume of paths between the two nodes. The more paths between the two nodes are, the higher the transition probability will be. Random walks fall into two parts: forward random walks and backward random walks. The forward random walks find the probability of ending at node $j$ when a t-step random walks starts at node $k$, denoted as $P_{t|0}(kj)$. The backward random walks find the probability of starting at node $k$ when a t-step walk ended at node $j$, denoted as $P_{0|t}(kj)$. There exists a rule where $P_{0|t}(kj) \propto P_{t|0}(jk) \cdot P_0(k)$. $P_0(k)$ is the probability of choosing $k$ as the starting node.

Combined with the $m$-Score of each member, we employ random walks on graph to derive the community tree from a social network. In experiments, we found that forward random walks perform better than backward random walks. Thus, the forward random walks is employed in our algorithm to derive the community tree.

Algorithm 2 Deriving Community Tree: CT_Deriving
Input: social network $G$
Output: Community Tree $CT$
Steps:
1. $CT \leftarrow [null, \cdots, null]$
2. $A \leftarrow getOneStepTransMatrix(G)$
3. $Z \leftarrow$ a diagonal matrix satisfied $Z_{ij} = \sum_i [A^t]_{ij}$
4. $M^t \leftarrow A^t \cdot Z^{-1}$
5. $R \leftarrow calcScore(G)$
6. For each $p_i$ in $R$
   7. $k \leftarrow argmax M^t[i][j]$
8. if $R[k] > R[i]$
9. $CT[i] \leftarrow k$
10. End
11. Return $CT$

Function $getOneStepTransMatrix(G)$ in step 2 calculates the one-step transition probability for each pair of nodes in a social network, and then organizes the one-step transition probabilities as a matrix $A$, which is a row stochastic matrix satisfying $\sum_j A_{ij} = 1$. $A[j,k]$ is the one-step transition probability from node $j$ to $k$.

$$P_{t+1|t}(k|j) = \begin{cases} (1-s) \cdot C_{jk} / \sum_i C_{ji}, & \forall k \neq j \\ s, & k = j \end{cases}$$

where $s$ denotes the self-transition probability, and $C_{ji}$ denotes weight of edge between node $j$ and $i$.

We derive a t-step transition probabilities matrix $M^t = A(A \cdots (A))$, and normalize $M^t$ according to $M^t = A_t \cdot Z^{-1}$, where $Z$ is a diagonal matrix. With $M^t$, we can find the most possible ending node $j$ when starting a t-steps forward random walks at node $i$. If $m$-Score of node $j$ is much greater than that of node $i$, we regard $j$ as the parent node of node $i$. After all operations end, we will obtain a community tree $CT$ represented in an array where $CT[i]$ indicates the parent node of node $i$. If the value in $CT[i]$ is null, it means that we cannot find the immediate leader of node $i$. Hence node $i$ is the core of community.

Node $i$ and its descendents compose a community. The $m$-Score of each member shows its importance in the community. And descendents of a node present its power scope. Parent node of each node is its immediate leader.

IV. ANALYZING EVOLUTION OF THE ORGANIZATIONAL STRUCTURE

In Section III, we proposed CT_Deriving algorithms to derive a community tree from a static social network, called static community tree. However, the static community tree can not well present the evolution of organizational structure in dynamic social networks.

To present evolution of organizational structure in a dynamic social network, we will aggregate the change of a social network in one time period, and then derive an evolving community tree from the community tree in the previous period and the one in current period. This enables a smooth change between the two community trees. Hence, the evolving community trees may well present evolution of organizational structure over time.

A. Scoring Function

To learn an evolving community tree, a scoring function is needed to evaluate the evolving community tree. Let $D(CT_i, CT_j)$ denote distance between two community trees $i$ and $j$. We use an error function $ES$ in (1), as a scoring function to measure the distance errors among three types of community trees: $CT_e$, $CT_p$, and $CT_c$, representing respectively the evolving one, the one in the previous period, and the one in current period. An optimal evolving community tree minimizes the value of $ES$.

$$ES = \sqrt{D(CT_e, CT_c)^2 + D(CT_p, CT_c)^2}$$

The tree edit distance [15] provides a measure of distance (similarity) between two trees, defined as the least cost of edit operation to change a tree to another tree. It can be employed to calculate $D(CT_i, CT_j)$.

B. Tree Learning Algorithm

In this paper, we propose a tree learning algorithm to find an evolving community tree from two static community trees, which is a process that reconstructed a community tree according to scoring function $ES$. It is uncertain that such an evolving community tree obtained from this process is a globally optimal solution. However, in each step of the reconstruction, we can derive a locally optimal one in the context. The reconstructing process is as follows.
1. Obtain a collection of members in the evolving community tree \( P_{ce} = P_{pre} \cup P_{cs} \) where \( P_{pre} \) and \( P_{cs} \) are collections of members in the static community tree of previous and current time periods respectively.

2. Compute the m-Score for each member \( p_i \) in the evolving community tree, \( m-Score(p_i) = (1 - \alpha) \cdot m-Score(p_i|pre) + \alpha \cdot m-Score(p_i|cs) \), where \( \alpha \) is smoothing factor. For those members that appear in the evolving community tree but not in current community tree, if their m-Score is less than a threshold \( \varepsilon \), we regard them as retired members and remove them from the evolving community tree.

3. On reconstructing the evolving community tree, we firstly generate a root node NULL for the tree. According to the definition of community tree, the m-Score of one node should not be larger than the m-Score of its parent node. Therefore, we pick up members in m-Score descent order. In each iteration, we put one member under each node in current evolving community tree respectively. Such a method can obtain a collection of candidate evolving community trees. We choose the minimum ES score candidate as a solution in each iteration. After all members are examined, an evolving community tree can be generated.

**Algorithm 3 Learning evolving community tree:**

\textbf{ECT\_Learning}\newline
Input: Community Tree \( CT_{pre}, CT_{cs} \)\newline
Output: Evolving community tree \( CT_e \)\newline

\textbf{Steps:}\newline
1. \( P_{ce} \leftarrow P_{pre} \cup P_{cs} \)\newline
2. For each \( p_i \) in \( P_{ce} \)\newline
3. \( m-Score(p_i) \leftarrow (1 - \alpha) \cdot m-Score(p_i|pre) + \alpha \cdot m-Score(p_i|cs) \)\newline
4. End\newline
5. \( P_{ce} \leftarrow P_{ce} - \{ p_i \in P_{ce} - P_{cs} | m-Score(P_i) < \varepsilon \} \)\newline
6. \( CT_e \leftarrow \text{build a community tree with only a root node} \)\newline
7. For each \( p_i \in P_{ce} \) in m-Score descent order\newline
8. \( CS \leftarrow \text{put } p_i \text{ under each node in } CT_e \text{ to generate a candidates collection} \)\newline
9. \( CT_i \leftarrow \arg\min_{CT_i \in CS} ES(CT_i) \)\newline
10. if \( ES(CT_i) < ES(CT_e) \) \( CT_e \leftarrow CT_i \)\newline
11. End\newline
12. Return \( CT_e \)

After derive a series of evolving community trees, we can present the evolution of the organizational structure in a dynamic social network.

V. EXPERIMENTS

In this section, we evaluate the capability of the proposed approach on discovering organizational structure and to exploring evolution of organizational structure in a dynamic social network. The approach is implemented in Java. The experiments were conducted on a 1.8GHz PC with Dual-Core Pentium 4 processor with 1GB RAM.

A. Datasets

We evaluate the proposed approach on two datasets: karate club dataset [17] and the known Enron email corpus. The karate dataset represents a relationship network at a university karat club which consisted of 34 members. We use the karate dataset to evaluate the performance of the proposed approach on finding organizational structure in static social networks. Enron email corpus may be converted into a social network in which a vertex denotes a user, an edge denotes communications between two users, and a weight on the edge denotes the number of communications. We preprocess the Enron dataset in following ways: mapping email addresses into users, and deleting duplicate emails. The original Enron data set contains 517,432 emails documents collected from 150 senior executives in Enron. After preprocessing, 148 unique users are identified. When only picking up email documents communicated among the 148 users, we collected 58,875 email records. In this paper, we use the preprocessed Enron dataset to evaluate the performance of the proposed approach on exploring evolution of organizational structure.

B. Experiments for finding organizational structure

We employ the karate dataset, shown in Figure 2(a), to evaluate \textit{CT\_Deriving}. The parameters for testing the algorithm include the steps of random walks \( steps=50 \), and self-transition probabilities \( p = \{0.85, 0.87, 0.92\} \). The organizational structure obtained from the karate dataset is illustrated in Figure 2(b), (c), and (d), where parameter \( p \) is set 0.85, 0.87 and 0.92 respectively. We can find that only one community is found when \( p = 0.85 \) where node 1 is the immediate leader of other fifteen nodes, but it is under the leadership of node 34. When \( p \) is increased to 0.87, we obtain two communities that are the same as the results in [9]. The organizational structure of communities is formed in a flat structure where node 1 and node 34 are leaders respectively. When \( p \) is increased to 0.92, the two communities are further expanded toward a hierarchical structure where node 25 and 26 are under the leadership of node 32, and node 7 supervises node 17. The experiment shows that increasing the self transition probability tends to explore the organizational structure from a flat structure toward a hierarchical structure.

C. The experiment for exploring the evolution of organizational structure in Enron dataset

This experiment applies the proposed approach on the Enron dataset to explore the evolution of organizational structure in the dataset. We firstly identify the associations between communities in different time periods. After threading these communities, we analyze how organizational structures evolve over time. Associations among communities are
categorized as splitting, merging, evolving, and emerging. The splitting indicates that cores of communities $ct$ and $ct_2$ in period $t$ are also members of community $ct_3$ in $t-1$ period. We say that $ct_3$ is split into $ct_1$ and $ct_2$. The merging indicates that the cores of community $ct_1$ and $ct_2$ in period $t-1$ are also members of community $ct_3$ in period $t$. The evolving indicates that the core of community $ct_1$ in period $t-1$ is the core of community $ct_2$ in period $t$. The emerging indicates that the core of community $ct_1$ in period $t$ did not appear in any community in $t-1$ period.

We derive twelve social networks from the Enron dataset by aggregating communications among users in every month in 2001, and then obtain twelve static community trees. We employ $ECT_{\text{Learning}}$ to generate eleven evolving community trees with the smoothing parameter $\alpha = 0.5$ and the threshold parameter $\varepsilon = 0.01$.

The graph in Figure 3 presents the evolution of communities in the Enron dataset where each user is assigned an ID number. A node in the graph denotes a community and the ID number of each node represents the core of the community. We briefly treat the ID number of a core as the community. The graph show that community 20 split into three communities in February, and then merged with community 63 in June. Community 20 further split in July and August, and disappeared in September.

We count the numbers of communities found from the social network in each time period, and then present them in Figure 4. Curve static and dynamic show the numbers of both static and evolving communities respectively. We compute the standard error for two curves, and obtain value 2.84 for static curve and value 1.42 for dynamic curve. Obviously, the stander error of dynamic is less than that of static. It clearly indicates that our proposed approach smoothed the change of communities.

By analyzing the evolution of organizational structure, we can observe the changes of importance of the members in communities over time. We use the number of its decedents to evaluate the importance of a node. Figure 5 shows the change of the importances of four users in Enron dataset, dasovich-j, watson-k, jones-t and williams-w3 (their ID numbers are 20, 138, 55, and 144 respectively).

By examining Figure 5, it can be seen that dasovich-j had a greater importance than that of other three users, but more fluctuating. The importance of dasovich-j reduced gradually since July. The importance of jones-t decreased since April and reached nearly to zero in May. The importance of williams-w3 and watson-k increased gradually since April and July respectively, and then both remained basically stable.

Combined with Figure 3, it can be seen that dasovich-j (ID number 20) was the core of community from January to August. But its importance began to reduce when dasovich-j was no longer the core in September.
VI. CONCLUSION

In this paper, we introduce the concept of organizational structure to represent the power of members and the scope of their power in a social network. We proposed a community tree to represent organizational structure in the social network and conceived an approach, which combines the PageRank algorithm and random walks on graph, to derive the community tree. In order to investigate the change of organizational structure in dynamic social network, we also propose a tree learning algorithm to derive evolving community trees, which smooths the transition between community trees over time. Experiments conducted on real datasets demonstrate the feasibility and applicability of the proposed approach.

In learning evolving community tree, we employ tree edit distance as the scoring function. However, the high complexity of the tree edit distance algorithm results in a great time overhead in exploring the evolution of organizational structure. In the future, we plan on improving efficiency of our approach.

REFERENCES


