User Interaction Based Community Detection in Online Social Networks

Abstract

Online social networks (OSNs) provide an excellent platform to connect, share, communicate, and interact among different groups of people. Discovering meaningful communities based on the interactions of different people in a social network is an active research topic in recent years. However, existing interaction based community detection techniques either rely on the content analysis or only consider the underlying structure of the social network graph, while identifying communities in OSNs. As a result, these approaches fail to identify *active communities*, i.e., communities based on actual interactions rather than mere friendship. To alleviate the limitations of existing approaches, we propose a novel solution of community detection in OSNs. The key idea of our approach comes from the following observations: (i) the degree of interaction between each pair of users can widely vary, which we term as *the strength of ties*, and (ii) for each pair of users, the interactions with mutual friends, which we term the *group behavior*, play an important role to determine their belongingness to the same community. Based on these two observations, we propose an efficient solution to detect communities in OSNs that utilizes the interactions of every pair of users as well as the interactions among mutual friends in a social network. The detailed experimental study shows that our proposed algorithm significantly outperforms state-of-the-art techniques for both real and synthetic datasets.

Keywords. social network analysis, community detection, link analysis, graph mining, data mining

1 Introduction

Online social networks (OSNs) have become the mainstream medium for communicating and socializing among friends, family members, colleagues, acquaintances, fans, activists, etc. The most popular OSNs (e.g., Facebook, Twitter, and Google+) attract hundreds of millions of users. Over one billion active users in Facebook, over five hundred million users in Twitter, more than five hundred million users in Google+ use these social networking sites to connect and communicate with each other, share personal and public information, and express their opinions. These interactions vary in types (e.g., messaging, wall posts, shares, photo tagging etc.) and in numbers (e.g., very active users *versus* inactive users), and provide a number of unique distinguishing characteristics of the users in a social network. The mammoth data associated with the users and their interactions with others form the basis of discovering many previously unknown relationships such as detecting communities in OSNs. In this paper, we exploit user interactions and propose a novel community detection algorithm for OSNs.

Community detection in social networks has gained a huge momentum in recent years due to its wide range of applications. These applications include online marketing, friend/news recommendations, load balancing, and influence analysis. The community detection involves grouping of similar users into clusters, where users in a group are strongly bonded with each other than the other members in the network. Most of the existing community detection algorithms assume a social network as a graph, where each user in the social network is represented as a vertex, and the connection or interaction between two users is depicted as an edge connecting two vertices. One can then apply graph clustering algorithms [1] to find different communities in OSNs.

An earlier body of research in this domain focuses on identifying communities based on the underlying structure (e.g., number of possible paths or the common neighbors between two vertices) of the social network graph. Popular approaches include maximal clique [2], minimum cut [3], modularity [4], and edge betweenness [5]. These methods use only the structural information (e.g., connectivity) of the network for community prediction and clustering. However, they do not consider the level/degree of interactions among users and thus fail to identify communities based on actual interactions rather than mere friendship, which we call *active communities* in this paper.

With the explosion of user generated contents in the social network, some recent approaches [6, 7, 8, 9] focus on exploiting the rich sets of information of the contents to detect meaningful communities in the network. Among these works, [6]

considers the contents associated with vertices, [8] considers the contents associated with edges, and [9] considers both vertex and edge contents for detecting communities. These works are mainly based on the content analysis that considers the semantic meaning of the contents while grouping users into different communities. Since these approaches analyze the contents to determine the communities, they are not suitable for real-time applications such as load balancing, which require online analysis. Moreover, none of these work considers the various interaction types available in OSNs and the degree of interactions among users, i.e., does not distinguish between the weak and strong links between the users.

We observe that the degree of user interactions can be vital in community detection in many applications that include load balancing and recommendation systems. For example, the astounding growth of social networks and highly interconnected nature among the end users make it a difficult problem to partition the load of managing a gigantic social network data among different commodity cheap servers in the cloud and make it scalable [10]. A random partitioning, which is the defacto standard in OSNs [11], will generate huge inter-server traffic for resolving many queries, especially, when two users with a high degree of interaction belong to two separate servers. However, if we are able to identify communities that have a high degree of interactions among the users within the community and low degree of interactions among inter-community users, place all users in a community in the same server, we can eliminate the inter-server communication to a great extent and at the same time can improve the query response time. Similarly, identifying communities based on the interactions of mutual friends can be an effective technique for predicting missing links between a pair of users and recommending friends based on those identified missing links.

In this paper, we propose a novel community detection technique that considers the structure of the social network and interactions among the users while detecting the communities. The key idea of our approach comes from the following observations: (i) the degree of interaction between each pair of users can widely vary, which we term as *the strength of ties*, and (ii) for each pair of users, the degree of interactions with common neighbors (e.g., mutual friends in Facebook), which we term the *group behavior*, play an important role to determine their belongingness to the same community. Based on these observations, we propose a community detection algorithm that identifies *active* communities into four phases. First, we model and quantify the interactions between every connected pair of users as the strength of ties, which allows us to differentiate strong and weak links in OSNs while detecting communities. Based on these interactions, in the second phase, we quantify the group behavior for every pair of users who are connected via common neighbors. Third, based on the interactions and group

behavior, we build a probability graph, where each edge is assigned a probability denoting the likelihood of two users to belong to the same community. Finally, we apply hierarchical clustering algorithm on the computed probability graph to identify communities. We conduct an extensive experimental study using a wide range of real and synthetic datasets. Experimental results show that our approach outperforms state-of-the art community detection algorithms in a wide range of evaluation metrics.

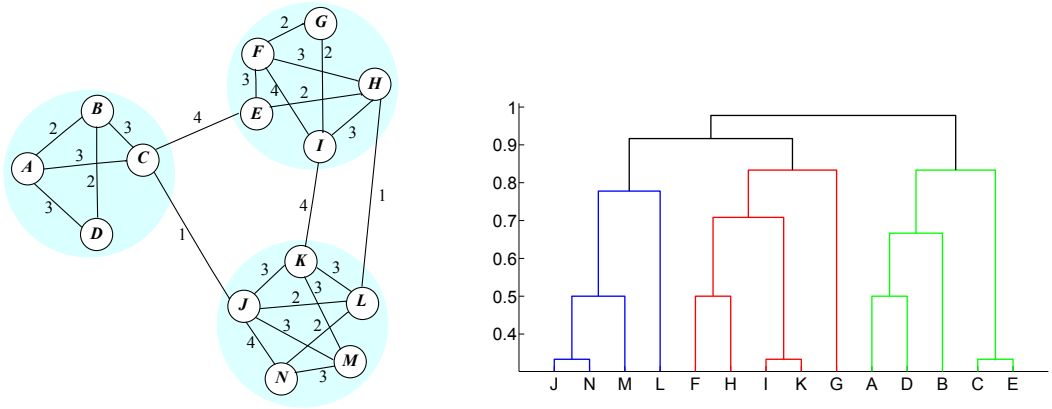
In summary, our contributions are as follows:

- We model and quantify the degree of interactions between users and the group behaviors among users.
- We propose a novel community detection algorithm for OSNs based on user interaction and group behavior to identify active communities with a high accuracy.
- We conduct extensive experiments using both real and synthetic datasets to show the effectiveness of our approach.

2 Problem Formulation and Preliminaries

A social network is represented using an undirected graph $G = (V, E)$, known as *social graph*, where the vertex set V represents users and the edge set E represents connections between users. If two users $u \in V$ and $v \in V$ are connected in the social network, then there exists an edge $e_{uv} \in E$ in the social graph G. There can be a weight w_{uv} associated with an edge $e_{uv} \in E$, which signifies the edge contents or interactions (e.g., number of messages, wall posts, etc.) between two users $u, v \in V$.

The community detection in a social graph G involves grouping similar vertices into clusters $C = \{C_1, C_2, C_3, ...\}$, where each cluster C_i contains vertices from V that are closely related, and hence forms a community. A popular school of thought to find communities from a social graph is to apply hierarchical clustering algorithms [3, 4, 12]. Primary disadvantages of hierarchical algorithms that consider edges having equal weights are: (i) they cannot capture true distance measure and thereby violate the triangle inequality, and (ii) they randomly start combining two vertices that are connected by an edge. These result in poor formation of communities. Even if a hierarchical algorithm [4, 12] assumes a *weighted* social graph, it leads to inappropriate communities as it considers merging clusters in presence of high weighted edges without considering whether the edge is an



(a) An Example Graph

(b) Standard Hierarchical Clustering

Figure 1: Standard Hierarchical Clustering of a Social Graph

inter-community edge, and thus wrongly groups two vertices from different communities into the same community. Figure 1(a) shows a social graph (weighted) and Figure 1(b) shows the resulting communities of the social graph using a hierarchical algorithm. Figure 1(a) shows a social graph of 14 vertices, where three real communities are shown using three big shaded circles. Figure 1(b) shows that vertex E forms a community with the vertices $\{A,B,C,D\}$, though it should appear in the community $\{F,G,H,I\}$. Similarly, vertex K forms a community with $\{F,G,H,I\}$ instead of appearing in the real community $\{J,L,M,N\}$. To alleviate the above limitations, we exploit both the degree of interactions, i.e., strength of ties between each pair of users, and the group behavior, i.e., interactions with common neighbors, while detecting communities in a social graph.

3 Related Work

With the recent advances in information networks, the problem of community detection has been widely studied in recent years. Many approaches formulated the problem in terms of dense region identification such as dense clique detection [2] [13] [14] and structural density [15] [16]. Further, there have been approaches formulating the problem in terms of minimum cut [3] and label propagation [17] [18]. However, commonly used methods of community detection involve modularity [4] [19] [20] and edge betweenness [4].

Among the aforementioned approaches, SCAN (Structural Clustering Algorithm

for Networks) [15] and Truss [16] use neighborhood concept of common neighbors, which has similarity with our group interaction concept for un-weighted graphs. However, the limitation associated with the aforementioned approaches is, most of these methods are pure link-based methods based on topological structures. They focus only on the information regarding the linkage behavior (connection) for the purposes of community prediction and clustering. They do not utilize different attributes present in networks and as a result their performance degrades in networks with rich contents, e.g., OSNs. In case of OSNs, these methods do not consider the various interaction types and degree of interactions among users, and hence fail to identify *active communities*.

The limitations of traditional methods along with the availability of rich contents in OSNs have led to the emergence of content based community detection methods [9] [8] [21] [6]. There is a growing body of literature addressing the issue of utilizing the rich contents available in OSNs to determine communities. Some recent works [21] [6] have shown the use of vertex content in improving the quality of the communities. Zhou et al. [6] combined vertex attributes and structural property to cluster graphs. Yang et al. [21] proposed a discriminative model of combining link and content analysis for community detection. However, these methods have been criticized [9] as certain characteristics of communities can not be modeled by vertex content. Again, there have been some recent proposals involving the use of edge content [9] [8] [7]. The Community-Author-Recipient-Topic (CART) model [22] is one of the first attempts to combine link based community discovery with contents. Qi et al. [9] integrated the structural and content aspects of network with the use of matrix factorization. Sachan et al. [8] used content and interaction to discover communities. These methods focus on utilizing edge content in the form of texts/images to discover communities. One major bottleneck of such content based community detection methods is their scalability. Content based community detection methods require analyzing rich textual/image attributes of OSNs which makes them unsuitable for real-time community detection from large-scale data.

4 Community Detection

In this section, we propose a community detection algorithm for OSNs based on interactions among users that range from simple messaging to participating together in different types of applications like games, location-based services and video chat. More specifically, to identify whether two social network users fall into the same community, our algorithm considers both the impact of interaction between them and the impact of the group behavior that the users show based on

the interaction with their common friends (i.e., common neighbors in the social graph). The key idea of our algorithm is to combine these impacts in a weighted manner and vary the weights to identify the active communities with a high accuracy.

The proposed community detection algorithm has four phases. In the first phase, the algorithm quantifies the degree of interaction between every connected pair of users in the OSNs and based on these interactions, in the second phase, the algorithm quantifies the group behavior for every pair of users who are connected via common neighbors. In the third phase, the algorithm determines the probability of two users belonging to the same community using the impact of interaction between them and their group behavior. Finally, in the fourth phase, the algorithm applies hierarchical clustering to detect communities based on the computed probabilistic measure.

4.1 Quantifying the Interaction

In this phase, the algorithm computes the degree of interaction between every connected pair of users in a social network. Given a social graph $G(V,E)$ and user interaction data, the algorithm constructs an *Interaction Graph*, $G_I(V,E,W)$, where each weight $w_{uv} \in W$ quantifies the degree of interaction between two users u and v. The higher the value of w_{uv}, the higher is the strength of tie between u and v. We have considered the following three factors to quantify interaction between two users: interaction type, average number of interactions for a particular interaction type, and relative interaction.

Interaction Type: Now-a-days OSNs involve interactions of different types. For example, Facebook users can interact via personal messages, wall posts, photo tags, page likes etc. To quantify the degree of interaction between two users, it is necessary to consider all interaction types. In addition, we observe that, some of these interaction types indicate stronger bonding than the others. Thus, it is important to prioritize the interaction types in an order. Prioritizing the interaction types in terms of bonding is especially useful for applications such as friend recommendation, and influence analysis. The prioritization can also be done considering the data transfer overhead associated with each interaction type, which is useful for applications like load balancing. We prioritize different interaction types using weights.

In addition, in an OSN, there could be both active and passive users. Passive users establish friendship with others, but hardly interact with them. Sometimes there is no interaction involved in a link established by a passive user. This can

also happen for newly joined users of an OSN. To determine the community in which a passive/new user belongs to, we consider the user's connection/link with others. More specifically, we consider the establishment of friendship between two users as a special type of interaction and to incorporate this special type of interaction, our communication detection algorithm provides a threshold value for each established friendship link.

Average Number of Interactions for a Particular Interaction Type: Another important factor to consider is the average number of interactions for a particular interaction type, which is not same for all interaction types. For example, average number of personal messages in Facebook is higher compared to average number of wall posts. To address this issue, we normalize the number of interactions for each type using the average value corresponding to the type. Otherwise, interaction types with higher average values eliminate the effect of type prioritization.

Relative Interaction: We observe that the importance of an interaction can vary among users based on their activities. To incorporate this issue in quantifying the interaction between users for our community detection algorithm, we take relative interaction into account. For ease of understanding, first consider an example in an un-weighted graph shown in Figure 2(left): Both users B and E are connected to user A and the number of connections of users B, E, and A are 2, 4, and 4, respectively. Thus, we observe that the friendship link AB has high importance to user B than user A as it represents 50% and 25% of activities of B and A, respectively. Further, the link AB is more significant than the link AE as the number of links associated with both A and E are higher than that of B.

Figure 2: Relative Intercation

The relativity of interaction also applies to weighted graphs. In Figure 2(right), we find that E is a highly active user with total interaction weight 21 and B is the least active user with total interaction weight 4. User A shows moderate level of activities with total interaction weight 15. We observe that AB involves 13% and 50% of total interactions of A and B, respectively and AE involves 27% and 19% of total interactions of A and E, respectively. Thus, interactions associated with AB seem more significant compared to interactions associated with AE.

To quantify the impact of relative interaction, we normalize each interaction in terms of total interaction of the involved users.

Next, we formally quantify the interaction between two users based on the above three factors.

Quantification: Let $\{I^1, I^2, ..., I^t\}$ represent t interaction types in an OSN. To prioritize the interaction types, we associate a weight with every interaction type. Assume that weights $W^1, W^2, ..., W^t$ are associated with $I^1, I^2, ..., I^t$, respectively, where $W^i > W^j$, if I^i represents stronger bonding between users than that of I^j. Weights can be predefined by experts or even by social network users. In our experiments, we conduct a survey among social network users to determine the weights. Note that for a particular type of social network, weights can be set once and used multiple times to identify communities in that social network. We first quantify interaction between users based on $\{I^1, I^2, ..., I^t\}$ and then to incorporate the impact of links without interaction, we add an additional threshold value, ε, to the quantified interaction.

Let i^t_{uv} represent the number of interactions of type t between users u and v, and n is the number of users in the social graph $G(V,E)$. The average number of interactions for a particular type t, $\bar{I}^t$, is computed as $\sum_{u \in V} \sum_{v \in V \wedge u \neq v} i^t_{uv}/n$. The normalized number of interactions of type t between u and v, I^t_{uv}, is computed as $i^t_{uv}/\bar{I}^t$.

Considering prioritized interaction types, links without interaction, and the average number of interactions for each interaction type, the algorithm quantifies interaction between two users u and v as $\hat{w_{uv}}$ using the following equation:

$$\hat{w_{uv}} = I^1_{uv} \times W^1 + I^2_{uv} \times W^2 + ... + I^t_{uv} \times W^t + \varepsilon \tag{1}$$

Finally, to incorporate the impact of the relative importance of an interaction between users u and v, the algorithm considers the quantified interactions of u and v with their neighbors $N(u)$ and $N(v)$ respectively (using Equation 1), and modifies $\hat{w_{uv}}$ as w_{uv} using the following equation:

$$w_{uv} = \frac{1}{2}\left(\frac{\hat{w_{uv}}}{\sum_{x \in N(u)} \hat{w_{ux}}} + \frac{\hat{w_{uv}}}{\sum_{y \in N(v)} \hat{w_{yv}}}\right) \tag{2}$$

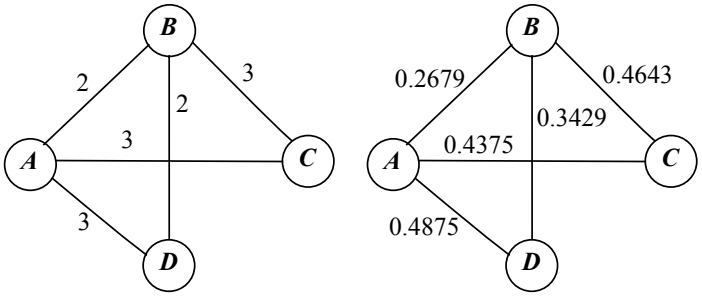
Without loss of generalization, consider a scenario, where Facebook users A, B, C, D interact using the public interactions shown in Table 1.

Let the weights for interaction type wall posts, photo tags and page likes be $0.4, 0.3, 0.2$, respectively and $\varepsilon = 0.1$. The average number of wall posts, photo tags and page likes for all users are 3, 9, and 38, respectively. Then the weight w_{AB} between A and B in the interaction graph is computed as follows:

Table 1: Interaction among A, B, C, D

Interaction Type	A,B	A,C	A,D	B,C	B,D
Wall Posts	12	15	12	9	9
Photo Tags	9	27	27	45	9
Page Likes	0	0	76	38	76

$$\hat{w_{AB}} = \frac{12}{3} * 0.4 + \frac{9}{9} * 0.3 + \frac{0}{38} * 0.2 + 0.1 = 2, w_{AB} = \frac{\frac{2}{2+3+3} + \frac{2}{2+3+2}}{2} = 0.2679$$



(a) $\hat{G_I}(V,E,\hat{W})$ (b) $G_I(V,E,W)$

Figure 3: Interaction Graph $G_I(V,E,W)$ Construction

By computing edge weights in the aforementioned way, we generate the interaction graph G_I as shown in Figure 3(b). Note that, the weight corresponding to every edge of interaction graph, w_{uv} lies between 0 and 1.

4.2 Quantifying Group Behavior

In this phase, the algorithm computes the group behavior that every pair of users, who are connected via one or more common neighbors, show in a social network. Any two users of a social community usually have common neighbors (mutual friends). Most of these common neighbors are other members of that community who interact with both of them. Thus, the algorithm quantifies the group behavior of pair of users based on the number of common neighbors and their interactions

with common neighbors.

Given an interaction graph, $G_I(V,E,W)$, the algorithm constructs a *Group Interaction Graph*, $G_{GI}(V,E',W')$, where there is an edge, $e'_{uv} \in E'$ between two users u and v, if they have at least a common neighbor. W' is a set of weights, where each weight quantifies the group behavior between two users with respect to common neighbors. Let $M_{uv} = \{m_1, m_2, ..., m_h\}$ represent h common neighbors (mutual friends) of u and v, where $m_1, m_2, ..., m_h, u, v \in V$. The interaction with a common neighbor $m_i \in M_{uv}$, $w_{uv}^{m_i}$ is quantified as $w_{uv}^{m_i} = \min(w_{um_i}, w_{vm_i})$.

The proposed algorithm computes the interaction of every pair of users who are connected via one or more common neighbors with respect to each of their common neighbors and then quantifies the group behavior for pair of users, $w_{M_{uv}} \in W'$ as follows:

$$w_{M_{uv}} = \sum_{i=1}^{h} w_{uv}^{m_i} \tag{3}$$

Consider the interaction graph G_I in Figure 3(b): A and B have two common neighbors C and D. The weights representing interactions of C with A and B are 0.4375 and 0.4643 respectively. Hence, the group behavior corresponding to A and B involving C is $w_{AB}^{C} = min(0.4375, 0.4643) = 0.4375$. Again, the interaction of D with A and B weights 0.4875 and 0.3429, respectively. Hence, the group behavior corresponding to A and B involving D is $w_{AB}^{D} = min(0.4875, 0.3429) = 0.3429$. Thus, the group behavior corresponding to A and B involving all common neighbors, $w_{M_{AB}}$ is computed as follows:

$$w_{M_{AB}} = w_{AB}^{C} + w_{AB}^{D} = 0.4375 + 0.3429 = 0.7804$$

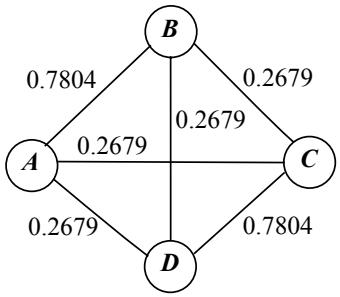


Figure 4: Group Interaction Graph $G_{GI}(V,E',W')$ Construction

By computing edge weights in the aforementioned way, we construct the group interaction graph G_{GI} as shown in Figure 4. Note that, the weight corresponding to every edge of group interaction graph, $w_{M_{uv}}$ lies between 0 and $(1 - w_{uv})$.

4.3 Computing the Probabilities

Given an interaction graph, $G_I(V,E,W)$, and a group interaction graph, $G_{GI}(V,E',W')$, the algorithm constructs a *Probability Graph*, $G_p(V,E \cup E',P)$, where each weight $p_{uv} \in P$ represents a probability between two vertices u and v to belong in the same community. The probability of two users sharing the same community depends on: the interaction between them and their group behavior. The group behavior between two users is determined in terms of their number of common neighbors and the interaction with them. If two social network users interact with a large number of common neighbors, it is very likely that they belong to the same community. The weights in W and W' represent user interaction and group behavior, respectively.

Formally, we define *probability* p_{uv} of two users u and v belonging to the same community as follows:

Definition 4.1 ***(Probability of belonging to the same community)*** *Let w_{uv} and $w_{M_{uv}}$ represent the weight of interaction between u and v and their group behavior, respectively, and α be a parameter used to combine the impact of w_{uv} and $w_{M_{uv}}$, where $0 \leq \alpha \leq 1$. The probability of u and v to belong to the same community, p_{uv}, is defined as follows:*

$$p_{uv} = \alpha * w_{uv} + (1-\alpha) * w_{M_{uv}} \tag{4}$$

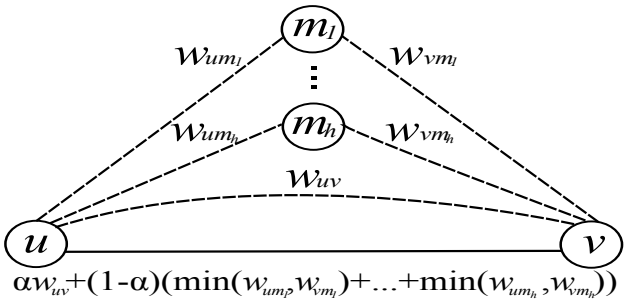


Figure 5: Probability (p_{uv}) Calculation

Here, both α and w_{uv} lie between 0 and 1. Further, $w_{M_{uv}}$ lies between 0 and $(1-w_{uv})$. Hence, the p_{uv} values generated by equation (4) lie between 0 and 1. We use the parameter α to control the impact of w_{uv} and $w_{M_{uv}}$ on probability for users u and v. We experimentally find the appropriate value of α that identifies the communities with a high accuracy.

Consider the interaction graph G_I in Figure 3(b) and the group interaction graph G_{GI} in Figure 4, where the interaction and group behavior corresponding to A, B, C, D are quantified. From G_I, we find that the quantified interaction between A and B, w_{AB}, is 0.2679 and from G_{GI}, we find that the quantified group behavior between A and B, $w_{M_{AB}}$, is 0.7804. The probability p_{AB} of A and B to belong to the same community is computed as follows:

$$p_{AB} = \alpha * w_{AB} + (1-\alpha) * w_{M_{AB}} = \alpha * 0.2679 + (1-\alpha) * 0.7804$$

For $\alpha = 0.5$, the probability of A and B belonging to the same community is 0.5*0.2679+0.5*0.7804, i.e., 0.5241.

By computing edge weights in the aforementioned way, we generate the probability graph G_p as shown in Figure 6. Note that, the weight corresponding to every edge of probability graph, p_{uv} lies between 0 and 1.

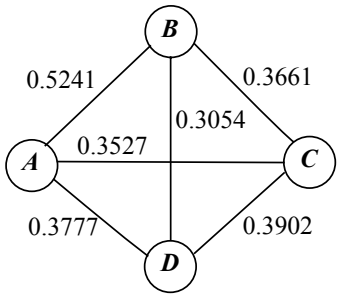


Figure 6: Probability Graph $G_p(V, E \cup E', P)$ Construction

.

4.4 Hierarchical Clustering on Probability Graph

The final phase of the proposed algorithm involves identifying communities by applying hierarchical clustering in the probability graph $G_p(V, E \bigcup E', P)$. We use agglomerative hierarchical clustering to build a hierarchy of clusters. By horizontally cutting the agglomerative hierarchical cluster tree at an appropriate height, we will get the required clusters (communities) $\{C_1, C_2, \ldots\}$.

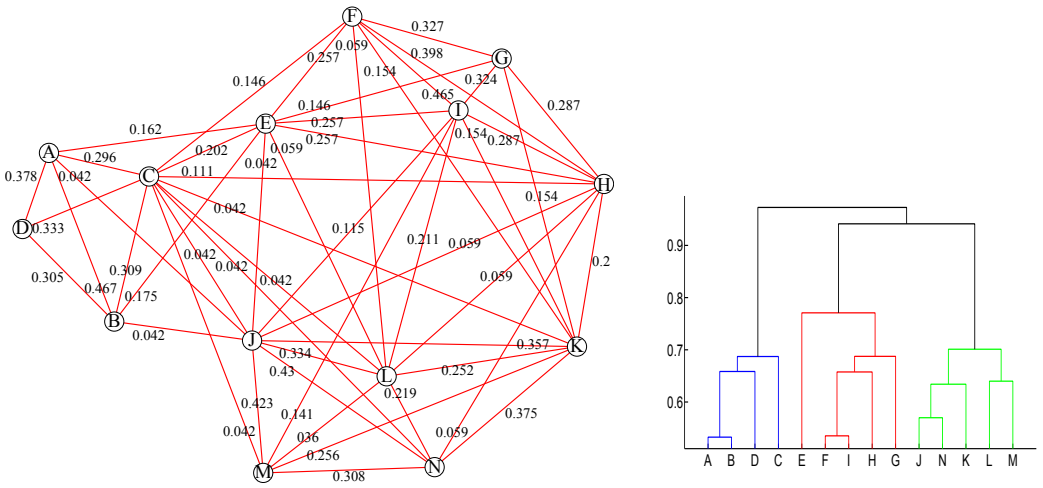
The hierarchical clustering algorithm works as follows. It starts with every node $v \in V$ in a separate cluster. At every iteration, larger clusters are constructed by combining two smaller clusters. This merging of clusters is done by first calculating the distance between every pair of clusters using the given *distance measure*

and *linkage criterion* and then merging two clusters with the shortest distance. The merging process stops when there is one cluster left. We use the following distance measure and linkage criterion for our hierarchical algorithm.

Distance Measure: In our approach, the probability $p_{uv} \in P$ of u and v to belong to the same community serves as the similarity measure. By subtracting p_{uv} from 1, i.e., $1 - p_{uv}$, we can have the distance between u and v.

Linkage Criterion: To estimate the distance between two clusters C_i and C_j, we have used 'Unweighted Pair Group Method with Arithmetic Mean (UPGMA)'as the linkage criterion. According to UPGMA, the distance is computed between every pair of users $u \in C_i$ and $v \in C_j$. Then the distance between any two clusters C_i and C_j is computed as the average of all computed distances:

$$\frac{1}{|C_i|.|C_j|} \sum_{u \in C_i} \sum_{v \in C_j} (1 - p_{uv})$$



(a) Probability Graph

(b) Clustering of Probability Graph

Figure 7: Clustering of the Example Graph

Figure 7(a) shows a probability graph and Figure 7(b) shows the hierarchical clustering on the probability graph. Note that, the probability graph is computed based on the graph in Figure 1. We observe from Figure 7(b) that, as a result of considering the group behavior, all the vertices (including E and K) are properly placed in the communities.

The clustering of the probability graph starts by merging the clusters $\{A\}$ and $\{B\}$. This is done as $\{A\}$ and $\{B\}$ have the highest probability of being together, i.e., they have the shortest distance between them. The corresponding probability value (p_{AB}) is 0.467. The next pair of clusters to be merged are $\{F\}$ and $\{I\}$. The corresponding probability value (p_{FI} is 0.465. Similarly, clusters $\{J\}$ and $\{N\}$ are merged. The next pair of clusters to be merged are $\{J,N\}$ and $\{K\}$. The probability value associated with these two clusters forming a community is the average of p_{JK} and p_{NK}, i.e., $(0.357+0.375)/2 = 0.366$.

4.5 Algorithm

Algorithm 1, *Detect_Communities*, shows the pseudocode for our community detection algorithm. The input to the algorithm is a social graph $G(V,E)$ and the user interaction data in the form of interaction matrices $A^1, A^2, \ldots, A^t$, where A^t represents interaction matrix for the interaction of type t. The size of each interaction matrix is $|V| \times |V|$ and each cell of an interaction matrix A^t denotes the number of interactions of type t between a pair of users in V. The output of the algorithm is a set of clusters, where each cluster represents an active community.

Algorithm 1: DETECT_COMMUNITIES($G, A^1, A^2, \ldots, A^t$)

Input : A social graph $G(V,E)$, Interaction matrices $A^1, A^2, \ldots, A^t$
Output: A set of clusters $C = \{C_1, C_2, ...\}$

```
1.1  G_I ← Construct_Interaction_Graph(G, A^1, A^2, ..., A^t);
1.2  G_GI ← Construct_Group_Interaction_Graph(G_I);
1.3  currentOptimal ← 0;
1.4  for α ← 0 to 1 do
1.5      G_P ← Construct_Probability_Graph(G_I, G_GI, α);
1.6      C^T ← Hierarchical_Clustering(G_p);
1.7      for clusterNum ← 1 to |V| do
1.8          C' ← Get(C^T, clusterNum);
1.9          if f(C') > currentOptimal then
1.10             C ← C';
1.11             currentOptimal ← f(C');
1.12         clusterNum ← clusterNum + 1;
1.13     α ← α + stepsize;
```

The algorithm first constructs G_I and G_{GI} using functions *Construct_Interaction_Graph* and *Construct_Group_Interaction_Graph* (Lines 1.1-1.2). Then, the algorithm varies the value of α from 0 to 1, and for each α, the algorithm computes probability graph G_p (Line 1.5). Since hierarchical clustering is applied on the

probability graph, the clustering tree C^T also varies with different values of α (Line 1.6). For every α, the algorithm varies the number of clusters, *clusterNum* from 1 to $|V|$, and for every *clusterNum*, the algorithm computes the set of clusters as C' from C^T (Line 1.8). The variable *clusterNum* is used to select the height at which the cluster tree C^T is cut and generate the clusters accordingly. The goal of our algorithm is to determine the pair $< \alpha, clusterNum >$ that provides us with the best *quality of clusters*. In our algorithm, we measure the quality of the computed set of clusters C' using a function $f(C')$ that evaluates the quality of clusters based on different standards such as modularity (Please see Section 5 for details). For every pair $< \alpha, clusterNum >$, the algorithm checks whether the quality of the computed set of clusters C' is better than the current best set of clusters C. If yes, then the algorithm updates C with C', otherwise C remains unchanged (Lines 1.9-1.10).

4.6 Time Complexity

Function *Construct_Interaction_Graph* in Line 1.1 needs $O(m)$ time, where m is the total number of edges in network $G(V,E)$. Again, function *Construct_Group_Interaction_Graph* in Line 1.2 needs $O(n\Delta^2)$ time, where the average vertex degree of the social network $G(V,E)$ is Δ. Similarly, the complexity of the function *Construct_Probability_Graph* in Line 1.5 is $O(m+n\Delta^2)$ which is iterated for each value of α. However, the corresponding loop is iterated a constant number of times (typically 100). Further, time complexity corresponding to the efficient implementation of hierarchical clustering (UPGMA) in Line 1.6 is $O(n^2)$ [23] and time complexity corresponding to the loop in Line 1.7 is $O(mn)$, when the chosen evaluation function is modularity. So, the time complexity of the complete algorithm is $O(n\Delta^2 + n^2 + mn)$. Now, in a typical social graph $m > n$. Hence the time complexity of the complete algorithm can be considered as $O((m+\Delta^2)n)$.

4.7 Discussion

In this subsection, we discuss different issues related to our proposed community detection algorithm.

Diverse Data Format: Our proposed community detection algorithm is consistent with diverse format of social networking data. We can identify communities from raw social networking data as per the methodology described in phase one to phase four. Again, we can identify communities from a basic weighted/unweighted social graph by normalizing the edge weights in terms of user degree as per equation (2), and running our algorithms for subsequent phases. Note that, in case of un-weighted graphs, all the edges have same weight, customarily 1.

Application Specific Community Detection: As per our methodology, identified communities from raw social networking data can be application specific depending on the priorities (weights) we put on different interaction types. For example, to serve the purpose of friend recommendation, the interaction type prioritization should be done considering the bonding, and in case of load balancing, the prioritization should be done considering the data transfer overhead associated with different interaction types. Therefore, the communities which will be used for friend recommendation might be different from the ones which will be used for load balancing.

Application-wise Time Complexity: The overall time complexity of the complete algorithm is $O((m+\Delta^2)n)$. However, time complexity corresponding to different applications can be significantly better.

In case of friend recommendation, we do not need to identify communities explicitly, rather we can recommend new connections (friends) from the group interaction graph. More specifically, if a link does not exist in the original interaction graph, and there exists a high weighted link in the group interaction graph between the corresponding users, then we can recommend these users to establish a link in the actual social network. In this case, the time complexity reduces to $O(\Delta^2 n)$.

In case of load balancing, the number of clusters acts as an input, and we do not need to find the proper number of clusters. Therefore, there is no search involved in optimizing the number of clusters (communities). In this case, the time complexity reduces to $O(n^2)$.

Overlapping Community Detection: In real-world social networks, individuals can belong to multiple communities, e.g., family, friends, colleagues, and schoolmates [24]. Many applications require identifying all community tags associated with a user. For example, a social network user may need to automatically partition his/her friends into overlapping social groups to propagate only relevant information to different groups.

Though our community detection algorithm has primarily been designed for identifying disjoint communities, it is also possible to identify overlapping communities using a modified version of our algorithm. More specifically, after identifying the disjoint communities, for each user (node), we can determine the probabilities associated with the user (node) to belong to different communities (not only the current community of the user). To determine the probability of a user to belong

to a particular community (this need not be the current community of the user), we consider the average of probability values (from the probability graph G_p) corresponding to the user and each of the current members of the community.

Consider the example graph in Figure 1, its corresponding probability graph in Figure 7(a), and the associated communities (clusters) in Figure 7(b). We can see that, our algorithm has identified three disjoint communities $\{A,B,C,D\}$, $\{E,F,G,H,I\}$, and $\{J,K,L,M,N\}$ from the example graph. Now, we can calculate the probability of A to belong to these three communities. For example, the probability of A to belong to community $\{E,F,G,H,I\}$ can be computed as: $\frac{p_{AE}+p_{AF}+p_{AG}+p_{AH}+p_{AI}}{\sum_x p_{Ax}} = \frac{0.162}{0.467+0.296+0.378+0.162+0.042} = 0.12$.

5 Experimental Evaluation

In this section, we evaluate the performance of our proposed community detection algorithm using both real and synthetic datasets. We compare our approach with the state-of-the-art community detection algorithms that include Girvan Newman [5], walktrap [25], fast greedy [19], leading eigenvector [26], infomap [27], label propagation [17] and multi-level [28] method using a number of evaluation metrics that include normalized mutual information (NMI) [21], pairwise F-measure (PWF) [21], and modularity [4].

5.1 Datasets

We have used four categories of datasets in our experiments.

Facebook User Interactions Labeled Dataset: In the first category, we have collected 2421 interactions from 221 Facebook users. These interaction types involve common page likes, photo tags, and wall posts. The average number of interactions corresponding to the aforementioned interaction types are 38, 9, and 3, respectively.

Since, users may want to put more weight on a certain type of interactions than the others, we have conducted a survey among users (whose data have been collected) to assign a weight for each type of interactions that signifies the relative importance of that particular interaction type. Upon conducting the survey, we have found that user preferred relative weights are 0.15, 0.35 and 0.5 for common page likes, tagged photos and wall posts, respectively. These collected data are later labeled into different classes. These labels are used as ground-truth values to measure the quality of our proposed community detection algorithm.

Facebook-like Forums Weighted Un-labeled Dataset: In the second category, we have used publicly available dataset of a Facebook-like Forum [29]. The forum represents an interesting two-mode network among 899 users and 522 topics, with 7089 edges. A weight is assigned to the link of a user and a topic, based on the number of messages or characters that the user posted on that particular topic. We have used the weighted (weighted by number of messages) static two-mode version of the network. This dataset is not labeled, and thus no ground-truth is available.

Classical Un-weighted Datasets: To show the wide range of applicability of our proposed approach, in the third category, we have used publicly available dataset of Facebook Ego network [30] along with three popular classical datasets: Zachary Karate Club [31], Dolphins [32], Jazz Musicians by Gleiser and Danon [33]. These networks are both undirected and un-weighted. The Facebook Ego network has 347 nodes and 5038 edges, the Zakhary Karate Club network has 34 nodes and 78 edges, the Dolphins network has 62 nodes and 159 edges, the Jazz Musicians network has 198 nodes and 2742 edges.

Computer Generated Benchmark Datasets: In the fourth category, we have used computer-generated benchmark graphs [34] that account for the heterogeneity in terms of external/internal degree. The generated graphs vary in terms of the mixing parameter that corresponds to the average ratio of external degree/total degree. Thus, by varying the mixing parameter as 0.1, 0.2, 0.3, and 0.4, we have generated four sets of social graphs. Each graph contains 128 vertices and they form four groups (i.e., vertices are labeled) each having 32 vertices. The average degree of vertices in each graph is 16 and the maximum degree is 25.

5.2 Performance Metrics

Our datasets consist of both labeled data and un-labeled data. Since the labeled dataset has the ground-truth values, we use supervised metrics: normalized mutual information (NMI) and pairwise F-measure (PWF) to evaluate the performance of our community detection algorithm. On the other hand, for both labeled and un-labeled datasets we use the most common measurement metric, modularity to compare the performance of our approach with the other state-of-the-art techniques.

Normalized Mutual Information (NMI): Let, $C = \{C_1, C_2, ..., C_k\}$ be the true community structure, where C_i corresponds to the community containing all vertices labeled with $1 <= i <= k$. Let $C' = \{C'_1, C'_2, ..., C'_k\}$ be the community struc-

ture returned by a community detection algorithm, where C'_i corresponds to the community containing all vertices labeled with $1 <= i <= k$ by the algorithm. Then the mutual information between the two community structures is defined as follows.

$$MI(C,C') = \sum_{C_i,C'_j} p(C_i,C'_j) log(\frac{p(C_i,C'_j)}{p(C_i)p(C'_j)})$$

By using the above defined mutual information, we can define the normalized mutual information as follows.

$$NMI(C,C') = \frac{MI(C,C')}{max(H(C),H(C'))}$$

Here, $H(C)$ and $H(C')$ are entropies of community structures C and C'.

Pairwise F-measure: Let T denote the set of vertex pairs that have the same label, S denote the set of vertex pairs that are assigned to the same community by the algorithm, $|T|$ and $|S|$ denote the cardinality of T and S, respectively. The pairwise F-measure (PWF) is computed from the pairwise precision and recall and can be defined as follows.

$$precision = \frac{|S \cap T|}{|S|}, recall = \frac{|S \cap T|}{|T|}, PWF = \frac{2 * precision * recall}{precision + recall}$$

Modularity: The most popular measure of evaluating communities of a un-labeled social graph is the modularity proposed by Newman [4]. Modularity is the fraction of the edges that fall within the given groups minus the expected such fraction if edges were distributed at random. Modularity, Q, can be defined as follows.

$$Q = \frac{1}{2m} \sum_{ij} [A_{ij} - \frac{d_i d_j}{2m}] \delta(s_i, s_j)$$

Here, m denotes the number of edges corresponding to adjacency matrix A, d_i denotes the degree corresponding to vertex v_i, s_i denotes the community membership of vertex v_i and $\delta(s_i, s_j) = 1$ if $s_i = s_j$.

5.3 Experimental Results

Facebook User Interactions Labeled Dataset: In this set of experiments, we compare our approach with five existing state-of-the-art community detection techniques. The results are summarized in Table 2. We observe that our algorithm achieves the *highest* NMI (0.7946) and PWF (0.7903) values which are significantly higher compared to other algorithms. The modularity, which is the most common unsupervised evaluation metric, of our algorithm is 0.484, which

Table 2: Facebook User Interactions Labeled Dataset

Algorithm	NMI	Precision	Recall	PWF	Modularity
Girvan-Newman	0.4860	**0.9947**	0.2347	0.3798	0.043
Walktrap	0.7754	0.9339	0.6608	0.7739	0.4414
Leading Eigenvector	0.7279	0.9137	0.6382	0.7515	0.4254
Infomap	0.3535	0.4266	0.2297	0.2986	-0.03
Multi-level	0.6274	0.601	0.5229	0.5592	0.448
Proposed Algorithm	**0.7946**	0.9738	**0.6650**	**0.7903**	**0.484**

is higher than other competitive algorithms. So, our proposed algorithm outperforms traditional algorithms in terms of both supervised and unsupervised metrics for the real Facebook dataset.

Facebook-like Forums Weighted Un-labeled Dataset: This dataset is weighted, where the weight of an edge represents number of messages between two vertices. However, the dataset is not labeled, and thus no ground truth about existing communities is available for this dataset. For this un-labeled dataset, we measure the modularity achieved by each community detection algorithm. We summarize the results in Table 3. As the network is weighted and interactions in the network are like OSNs, our algorithm performs better in this dataset. We can see that our algorithm achieves modularity value of 0.315 which is better than the values of most competitive algorithms.

Table 3: Facebook-like Forums Weighted Un-Labeled Dataset

Algorithm	Modularity
Girvan Newman	0.0488
Walktrap	0.2031
Leading Eigenvector	0.1696
Infomap	0.1372
Multi-level	**0.3458**
Proposed Algorithm	0.315

Classical Un-weighted Datasets: These graphs are un-weighted and do not allow us to construct *interaction graph* as per our model. Hence, we assume equal weights (i.e., equal to one) for all connecting edges, normalize the edge weights in terms of user degree as per equation (2), and run our algorithms for subsequent phases. The results obtained for these datasets are summarized in Table 4. In Karate and Dolphins network, our algorithm performs better than other algorithms and achieves the *highest* modularity value of 0.419 and 0.524 respectively. How-

ever, in Jazz network, our algorithm performs moderate and achieves the modularity value 0.42 which is close to the maximum modularity value 0.44. Again, in Ego network our algorithm achieves the modularity value of 0.433 which is close to the maximum modularity value 0.442.

Table 4: Classical Un-weighted Datasets

Algorithm	Karate	Dolphin	Jazz	Ego
Girvan Newman	0.401	0.519	0.405	0.271
Walktrap	0.353	0.489	0.438	0.398
Fast Greedy	0.381	0.495	0.439	**0.442**
Leading Eigenvector	0.393	0.491	0.394	0.433
Infomap	0.402	0.523	0.28	0.418
Label Propagation	0.36	0.379	0.282	0.343
Multi-level	**0.419**	0.519	**0.44**	0.427
Proposed Algorithm	**0.419**	**0.524**	0.42	0.433

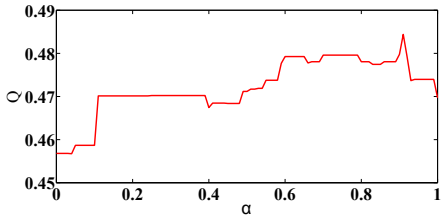
Computer Generated Benchmark Datasets: In the last set of experiments, we use computer-generated benchmark graphs to compare the effectiveness of our algorithm with other competitive methods. We have generated four graphs, namely, Dataset-1, Dataset-2, Dataset-3, and Dataset-4 for four different mixing parameter values 01, 0.2, 0.3, and 0.4, respectively. We have measured modularity performance of all algorithms for these graphs. The obtained results are summarized in Table 5. We can see that, in graphs with low connectivity outside clusters (mixing parameter values 0.1, 0.2 and 0.3), the modularity performance is almost identical for all algorithms. However, with an increased ratio of external degree/total degree (mixing parameter value 0.4), the modularity values of the most established algorithm degrades. We observe that our algorithm performs well under any external degree/total degree ratio.

Table 5: Computer Generated Benchmark Datasets

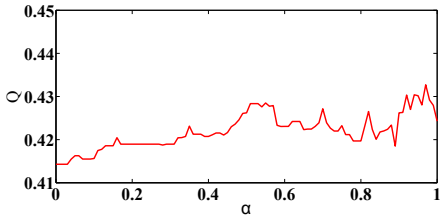
Algorithm	Dataset-1	Dataset-2	Dataset-3	Dataset-4
Girvan Newman	0.6533	0.5498	0.4541	0.341
Walktrap	0.6533	0.5498	0.4541	0.341
Fast Greedy	0.6533	0.5498	0.4541	0.2555
Leading Eigenvector	0.6533	0.4404	0.4541	0.3234
Infomap	0.6533	0.5498	0.4541	0.341
Label Propagation	0.6533	0.5498	0.4541	0.1797
Multi-level	0.6533	0.5498	0.4541	0.341
Proposed Algorithm	0.6533	0.5498	0.4541	**0.3427**

5.4 Value of α for Different Networks

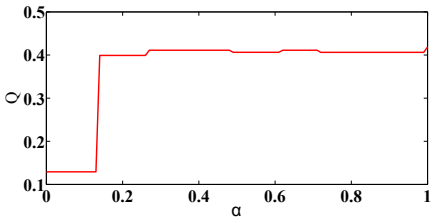
As we have discussed in Section 4, we can vary α to tune our algorithm to achieve a higher modularity value, which in turn means good quality communities. We vary α and see the effect of α in modularity (Q) for different datasets.



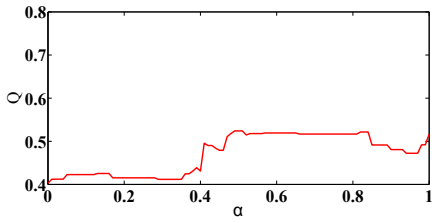
(a) Facebook Interaction Network



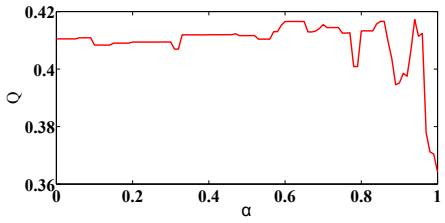
(b) Ego Network



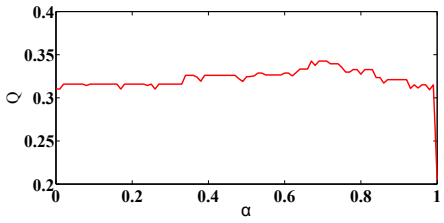
(c) Karate Network



(d) Dolphin Network



(e) Jazz Network



(f) Dataset-4

Figure 8: α vs Q for Different Networks

For networks which have relatively sparse connections within the group, i.e., the number of common neighbors for any two members of the group is relatively low, the algorithm achieves better modularity values for higher values of α i.e, one-to-one interaction gets priority. Karate network (Figure 8(c)) can be considered in this regard. It is to be noted that, in some cases completely eliminating

group interaction degrades modularity performance significantly. For example, Figure 8(e) shows that modularity value for Jazz Musicians dataset sharply falls, when α reaches to 1. Another interesting case appears for Dataset-4 (Figure 8(f)). We observe that when the network has almost balanced external/total degree ratio, we achieve the highest modularity value when α is near 0.5. These results validate our claim made in Section 4 that for different social network graphs the value of appropriate α can be different.

6 Work in Progress: Privacy Preserving Social Graphs for High Precision Community Detection

Discovering communities from a social network requires publishing the social network's data. However, community detection from raw data of a social network may reveal many sensitive information of the involved parties, e.g., how much a user is involved in which communities. An individual may not want to reveal such sensitive information. To resolve this issue, we address the problem of privacy preserving community detection in social networks. More specifically, we want to ensure that community detection is possible from the published social graph/data but the identity of users involved in a community should not be disclosed.

6.1 Background

Several anonymization methods have been proposed to battle the privacy attacks on social network data [35, 36, 37]. A popular class of these methods involves graph modification, i.e., anonymizing a graph by modifying (i.e, inserting and/or deleting) edges and vertices. However, none of these works strongly consider preserving *community structure* while developing privacy protection techniques. As a result, the developed graph modification techniques lead to indiscriminate modification of edges without focusing on the underlying community structure. Such indiscriminate modification of edges may disfigure the community structure and lead to misleading communities which highly deviate from the communities in the original social graph. Thus, state-of-the art graph modification approaches fail to serve our purpose of high precision community detection from privacy preserving social graphs.

6.2 Methodology

Our solution is based on the probability graph, where each edge is assigned a probability denoting the likelihood of two users to belong to the same community. We use the likelihood information from the probability graph to construct

a privacy-preserving version of the original social graph which is highly accurate in terms of community detection queries. In particular, we greedily modify the original social graph focusing on the community structure preservation, using the likelihood information from the probability graph.

Privacy Preserving Social Graph: We construct privacy preserving version of social graph which can be used to identify communities analogous to the communities in the original social graph. We construct the privacy graph by greedily modifying the original graph using likelihood information from the probability graph. The key idea of this greedy modification is to replace the random edge deletion-addition scheme of state-of-the-art graph modification approaches with a biased random scheme that favors the existing inter-community edges during edge deletion and the non-existing inner-community edges during edge addition.

Biased Random Scheme: Under the biased random scheme, (i) an existing potential inter-community edge has higher probability of being deleted and (ii) a non-existing potential inner-community edge has higher probability of being added, during the modification of the social graph to construct a privacy preserving version of it. The scheme identifies the potential inter and inner community edges based on the likelihood information from the probability graph G_p. Note that, a high weighted edge in the probability graph is a potential inner-community edge, and a low weighted edge in the probability graph is a potential inter-community edge. The scheme works as follows:

1. Split the edge set of probability graph G_p into two disjoint sets E (analogous to the edge set in G) and E' (potential inner-community edges non-existent in G). Note that, each edge e_{uv} that belongs to one of these sets, has an associated probability value p_{uv}.

2. Create an edge set E^p for the privacy graph and initialize it with the edges from the set E.

3. (i) For each edge $e_{uv} \in E$, calculate its probability of being deleted from the privacy graph as: $\frac{f(1-p_{uv})}{\sum_{e_{ij} \in E} f(1-p_{ij})}$. (ii) For each edge $e_{uv} \in E'$, calculate its probability of being added to the privacy graph as: $\frac{f(p_{uv})}{\sum_{e_{ij} \in E'} f(p_{ij})}$. Here, $f()$ is a monotonic function which defines the degree of bias during privacy graph construction. For example, an exponential $f()$ function implies a highly biased graph modification technique, which is most likely to delete only the existing inter-community edges and add only the non-existing inner-community edges during edge modification. However, such

a function makes the process relatively deterministic, which is not preferred in case of privacy preserving techniques.

4. Calculate the cumulative deletion/addition probabilities (cp) for edge set E/E'.

5. To delete/add an edge, (i) generate a random number $U(0,1)$, (ii) if the number is within the range $cp_{(i-1)}$ and cp_i, delete/add the ith edge of the set E/E' from/to the edge set E^p. Note that, cp_0 is 1.

6. For consecutive m edge deletion or addition, repeat step 3 to step 5 for m times.

During edge addition, one can also allow edges that are not in the set E' by preserving their slot. For example, to probabilistically allow $x\%$ edges barring the set E', we need to calculate the addition probability of an edge $e_{uv} \in E'$ (associated with step 3) as: $(1 - \frac{x}{100}) * \frac{f(p_{uv})}{\sum_{e_{ij} \in E'} f(p_{ij})}$.

Further, for the edges outside E', we need to distribute the remaining probability $(\frac{x}{100})$ uniformly among the edges.

The biased random scheme can also be used with weighted graphs. In case of weighted graphs, we assign weights to the newly added edges by generating random numbers using the probability distribution corresponding to the current edge weights.

6.3 Experimental Evaluation

We evaluate the performance of our proposed privacy graph construction method by comparing it with the state-of-the-art random m edge deletion-insertion method backed up by many approaches [38]. We compare these two methods in terms of the degree to which these privacy preserving methods preserve the underlying community structure of the original social graph. More specifically, for each of these privacy preserving methods, we determine the relevance/similarity of identified communities from the original social graph and the privacy graph constructed via the corresponding method. We use normalized mutual information (NMI) and pairwise F-measure (PWF) to compare the similarity of identified communities from the original social graph and corresponding privacy graph. Then, we compare the NMI and PWF values (degree of relevance) attained by each method to identify the superior method in terms of community structure preservation. We

can see that, for two networks (Karate and Jazz) our algorithm achieves significantly higher NMI and PWF values compared to the competing random *m* edge deletion-insertion method.

Algorithm ↓	Karate		Jazz	
	NMI	PWF	NMI	PWF
Proposed	0.8283	0.8551	0.8643	0.8539
Random *m* del-add	0.4759	0.4454	0.6753	0.5257

7 Conclusion

In this paper, we have proposed a community detection algorithm based on user interactions in online social networks. To identify active communities, we have considered both the impact of interaction between users and the impact of the group behavior that the users show based on the interaction with their common friends. We have combined both impacts in a weighted manner to identify the active communities with a high accuracy. To the best of our knowledge, this is the first approach that takes both pair interaction and group interaction into account for community detection. Our detailed experimental results show the superiority of our approach over other state-of-the-art techniques for a wide range of datasets. We have observed that for a real Facebook user interaction dataset, our algorithm achieves the *highest* normalized mutual information (NMI) and pairwise F-measure (PWF) values, which are significantly higher than the NMI and PWF values achieved by other competitive methods. Moreover, our algorithm performs reasonably well for different benchmark datasets, for both weighted and un-weighted social graphs.

Bibliography

[1] Aggarwal, C.C., Wang, H., eds.: Managing and Mining Graph Data. Volume 40 of Advances in Database Systems. Springer (2010)

[2] Abello, J., Resende, M.G., Sudarsky, S., Sudarsky, R.: Massive quasi-clique detection. In: In Latin American Theoretical Informatics (LATIN. (2002) 598–612

[3] Newman, M.E.J.: Detecting community structure in networks. The European Physical Journal B - Condensed Matter and Complex Systems **38** (2004)

[4] Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review **E 69** (2004)

[5] Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proceedings of the National Academy of Sciences **99**(12) (June 2002) 7821–7826

[6] Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. Proc. VLDB Endow. **2**(1) (2009) 718–729

[7] Zhou, W., Jin, H., Liu, Y.: Community discovery and profiling with social messages. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. (2012) 388–396

[8] Sachan, M., Contractor, D., Faruquie, T.A., Subramaniam, L.V.: Using content and interactions for discovering communities in social networks. In: Proceedings of the 21st international conference on World Wide Web. (2012) 331–340

[9] Qi, G.J., Aggarwal, C.C., Huang, T.S.: Community detection with edge content in social media networks. In: ICDE. (2012) 534–545

[10] Pujol, J.M., Erramilli, V., Siganos, G., 0001, X.Y., Laoutaris, N., Chhabra, P., Rodriguez, P.: The little engine(s) that could: Scaling online social networks. IEEE/ACM Trans. Netw. **20**(4) (2012) 1162–1175

[11] Rothschild, J.: High performance at massive scale - lessons learned at facebook. http://cns.ucsd.edu/lecturearchive09.shtml

[12] Newman, M.: Fast algorithm for detecting community structure in networks. Physical Review E **69** (2003)

[13] Zeng, Z., Wang, J., Zhou, L., Karypis, G.: Out-of-core coherent closed quasi-clique mining from large dense graph databases. ACM Trans. Database Syst. **32**(2) (June 2007)

[14] Pei, J., Jiang, D., Zhang, A.: On mining cross-graph quasi-cliques. In: Proceedings of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining. (2005)

[15] Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.J.: Scan: a structural clustering algorithm for networks. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. (2007) 824–833

[16] Cohen, J.: Trusses: Cohesive subgraphs for social network analysis. (2008)

[17] Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. (September 2007)

[18] Meng, W., Feng, L., Bressan, S., Winiwarter, W., Song, W., eds.: Database Systems for Advanced Applications, 18th International Conference, DASFAA 2013, Wuhan, China, April 22-25, 2013. Proceedings, Part II. In Meng, W., Feng, L., Bressan, S., Winiwarter, W., Song, W., eds.: DASFAA (2). Lecture Notes in Computer Science, Springer (2013)

[19] Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. **70**(6) (December 2004)

[20] Meo, P.D., Ferrara, E., Fiumara, G., Provetti, A.: Generalized louvain method for community detection in large networks. In: ISDA, IEEE (2011) 88–93

[21] Yang, T., Jin, R., Chi, Y., Zhu, S.: Combining link and content for community detection: a discriminative approach. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. (2009) 927–936

[22] Pathak, A.B.N., Erickson, K.: Social topic models for community extraction. In: The 2nd SNA-KDD Workshop 08 (SNA-KDD08), Las Vegas, Nevada, USA (August 24 2008)

[23] Mllner, D.: http://math.stanford.edu/ muellner/fastcluster.html

[24] Palla, G., Dernyi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature **435**(7043) (June 2005) 814–818

[25] Pons, P., Latapy, M.: Computing communities in large networks using random walks. J. of Graph Alg. and App. bf **10** (2004) 284–293

[26] Newman, M.E.J.: Finding community structure in networks using the eigenvectors of matrices. phys. Rev. E 036104

[27] Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. Proc. Natl. Acad. Sci. USA (2008) 1118

[28] Blondel, V., Guillaume, J., Lambiotte, R., Mech, E.: Fast unfolding of communities in large networks. J. Stat. Mech (2008) P10008

[29] Opsahl, T.: Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. Social Networks (August 2011)

[30] SNAP: Stanford network analysis project

[31] Zachary, W.: An information flow model for conflict and fission in small groups. Journal of Anthropological Research **33** (1977) 452–473

[32] Lusseau, D.: The emergent properties of a dolphin social network. Proceedings of the Royal Society of London. Series B: Biological Sciences **270**(Suppl 2) (November 2003)

[33] Gleiser, P.M., Danon, L.: Community structure in jazz. Advances in Complex Systems **6**(4) (2003) 565–574

[34] Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Phys. Rev. E **78**(4) (October 2008) 046110

[35] Zhou, B., Pei, J., Luk, W.: A brief survey on anonymization techniques for privacy preserving publishing of social network data. SIGKDD Explor. Newsl. **10**(2) (December 2008) 12–22

[36] Wu, X., Ying, X., Liu, K., Chen, L.: A survey of privacy-preservation of graphs and social networks. In: Managing and Mining Graph Data. Volume 40. (2010) 421–453

[37] Zheleva, E., Getoor, L.: Privacy in social networks: A survey. In: Social Network Data Analytics. (2011) 277–306

[38] Hay, M., Miklau, G., Jensen, D., Weis, P., Srivastava, S.: Anonymizing social networks. Technical Report 07-19, University of Massachusetts Amherst (March 2007)