DETECTION OF FRAUD GROUP IN ONLINE COMMUNITIES BY CLUSTERING ALGORITHM

By

Mostafizur Rahman & Mohammad Mojahidul Islam

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF SCIENCE
AT
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DHAKA 1000
2016

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical and Electronic Engineering for acceptance a thesis entitled "**Detection of Fraud Group in Online Communities by Clustering Algorithm**" by **Mostafizur Rahman & Mohammad Mojahidul Islam** in partial fulfillment of the requirements for the degree of **Bachelor Of Science**.

Dated: 2016

Supervisor: _____
Dr. S. M. Farhad

Readers: _____

_____

To all the people who works for open sources projects out of no personal benefit

Table of Contents

List of Tables

List of Figures

Abstract

Online communities have become very popular in the recent times. These communities have become very helpful in sharing knowledge, ideas, skills and even experiences among the users. Many people rely on the information of these online communities. It is important to make sure that the knowledge base of the online communities is not biased or corrupted. In this article, we have introduced the most common fraudulent activities found in the online communities . We have proposed a couple of measures and a graph based clustering algorithm to detect the most frequent fraudulent activities. The algorithm can detect the groups those are behind these fraudulent activities. We have implemented our approach and conducted experiments on two different data sets. The results show that our algorithm can fairly detect all the fraudulent activities we have introduced.

Chapter 1

Introduction

Online communities play an important role in sharing knowledge, ideas, skills even experience nowadays. In these communities people ask questions on different topics and others having previous knowledge or experience on that topic reply to those questions. Knowledge sharing culture has already grown up this way and it is gradually getting a shape. Millions of users around the world use regularly a number of **Question Answering (QA)** sites and web based discussion forums. It is important to make sure that the knowledge sharing people are not biased or corrupted.

1.1 Background

The major online communities include Stack Exchange Network, Yahoo Answers, Quora, Answers.com, Google Product Forum etc. Apart from these, there are also a number of other small to big communities(Question Answering or discussion forum) across the web. All of these communities share common features: awarding points, scores or other rewards to the users in recognition to their contributions to the community. When the contributions of a user to the community by solving others' problems or writing something to the community are found useful to other community

members gets some points or scores in recognition to his/her contribution. For instances, when a question or an answer is upvoted in StackOverflow the writer of that question or answer gets some points, in Quora the number of upvotes or downvotes are counted, in Yahoo Answer there is similar feature of point based score system like StackOverflow, in Answers.com they use trust points, Google Product forum uses plus one [22, 4]. Whatever the terms these communities use, they serve the same purpose to quantify the users' contributions to the community.

In this thesis we will analyze our algorithms with two different data sets. The first one is synthetic data which is created intentionally with some fraud groups. The second one is Durbin Community Application data source which is collected by an exclusive agreement with Durbin Labs who are the proprietor of this data.

1.2 Motiovation

There is no doubt that the online communities are playing a vital role in learning new things and sharing knowledge, building a massive knowledge base overall. But there is no guarantee that the answers someone has shared is absolutely correct. There must be strong logics and references to the answer to be a suitable one. For this reason awarding some points or other rewards like badges, medals and other credits to the answerer for the better answers has become a common culture to encourage the quality answers [36]. This has established a trend that, the more points and other rewards a person has, the more he knows. People believe that the person with higher rating points has more knowledge.

Like many other online forums StackOverflow is an online platform where the users can ask and answer questions regarding computer programming [20]. Any user can

ask a question describing the problem he is facing. If other users find the question as a good one they can appreciate the question by giving an upvote. This upvote adds some points to the user who asked the question. The questions asked in the forum are shown to all the users of the community and anyone can answer the question. If any user knows the solution to the problem described in the question can reply with details and references. If this reply solves the problem of the user who asked the question, he can accept the reply which rewards some points to the replier. Other users can also upvote the answer if they find the reply useful. The upvotes in answers adds some points to the answerer.

It is usual that the more skilled and experienced persons can provide better answers. The better answers get more upvotes and the answerer get more points. Consequently, it is expected that the users with more points are more skilled or experienced. This trend has direct effect on the hiring sectors of the technology industries. The recruiters believe that the candidate with higher rating are potentially better than others. This point system plays an important role in filtering or even selecting the candidates during hiring process.

The significance of the score or point system of the online communities is stated above. But there are many unfair means which are sometimes taken to increase the point of a specific user or a specific group of users. This fraudulent activities may ruin the acceptability of the point system and question the validity the knowledge base. For example, user A wants to increase his points. He opens a new account and starts voting up on all his answers. He can repeat this process i.e. create new accounts and vote up on each of his answers. This will increase his points, but this does not mean that user A has more experience or skills because much of his/her points are gained

through fraudulent actions.

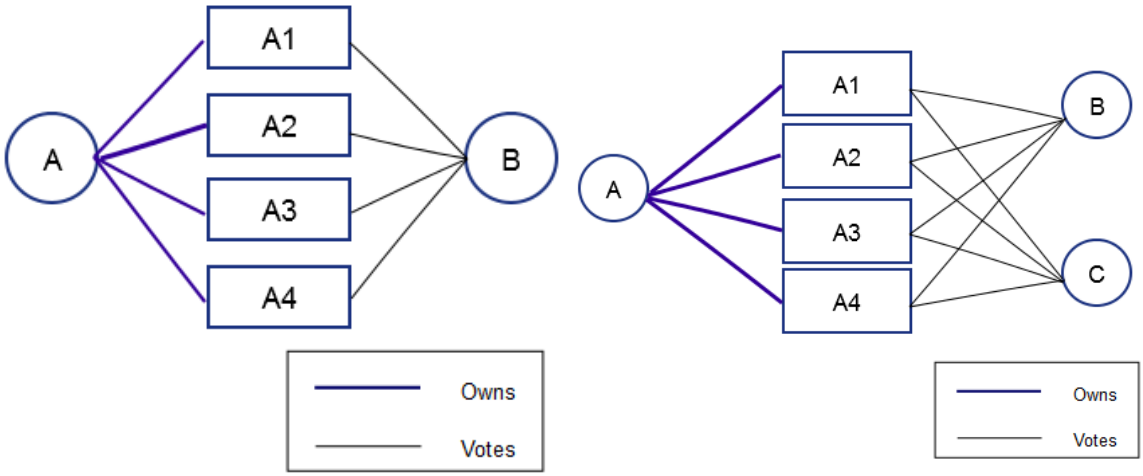
To keep the point system stable and trustworthy we need to make sure that neither knowledge base nor the rating system is not biased or corrupted. It is important to develop a detection system that can effectively detect the fraudulent activities in a rational manner.

1.3 Problem Definition

The main problem in the current scoring or point system is its vulnerability to the planned or intentional attacks. This weakness motivates us to design a solution that can predict the nature of the votes of users to other users' post (both question and answer). To do this we need to clarify the exact problems that persists in the current system. We have figured out the fundamental weakness of the current system and named these problems with specific title which will be used throughout this article.

Serial Upvoting is the first type of fraudulent activity. *Serial Upvoting* occurs when a specific user continuously upvotes to another specific user's post in a short interval period of time [32]. The reasons behind serial voting may be the alliance between two persons. In serial upvoting the first user serially upvotes the second user's post in a very short period of time. It is not necessary for the second user to vote back the first user as the either way itself is a fraudulent activity. This type of activities are practically the most common fraudulent activity in online communities.

Serial Downvoting is just the opposite of *Serial Upvoting*. In *Serial Downvoting* a specific user continuously downvotes to the posts of another specific user in a short period of time. The primary reason behind serial downvoting may be to take the



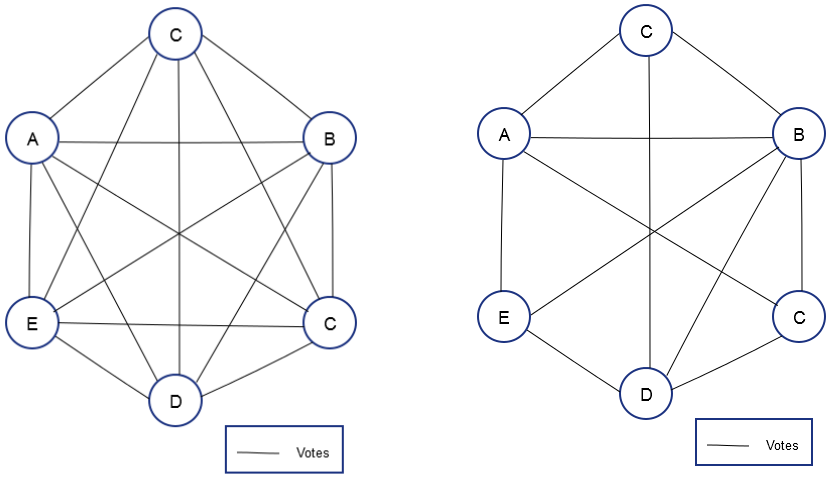
(a) (b)

Figure 1.1: (a) B is serially up/down voting answers of A (b) B and C are Sock Puppets of A

revenge of any previous issues between the users or any other private stuffs between themselves. This is not necessary to be two way, that is the other user does not necessarily need to vote back to the first user.

Sock Puppeting means pretending to be a real user. But it is not actually so. *Sock Puppeting* is found when a user run multiple account to support his activities of one account from others [41]. Finding out the puppet accounts is the most complex task here. Because the voting pattern a puppet used to follow may resemble with the voting pattern of a real account. We cannot classify the puppet account to be actually a puppet account with probability of 1 for this reason. Still there are some similarities among the activities of the puppet accounts. We have proposed a couple of measures to extract fraudulent activites and an algorithm to identify the puppet accounts and mutually collaborated voting groups.

Voting Group is a group of users who have mutually upvotes each others' posts.



(a) (b)

Figure 1.2: Examples of (a) Complete voting group and (b) Incomplete voting group

There is no particular voting pattern or voting order in this type of activity, but this is clear that the group members are casting votes in such a way that increases their points or scores. To figure out this type of activities, we have studied to find out how many *Voting Groups* are there and which user belongs to which group. Cases may also arise where a single user belongs to multiple groups.

Chapter 2

Related Works

In this chapter, we have discussed about the techniques and algorithms used in the literature of community or group detection in social networks. We have also discussed about the approaches that have been used in the recent years to detect fraud in different domains. These domains include detecting fraud in credit card transactions, online stack exchanges and different financial networks.

In many existing fraud detection systems, procedures of detecting fraud operates by adding fraudulent claims/transactions to "black lists" and then matches for likelihood the fraudness in the new claims/transactions. Some systems use straight forward hard-coded rules. Thus, to claim a transaction to be fraud each transaction should meet these rules, such as matching addresses and phone numbers, and price and amount limits.

Size of both fraud and legal classes will fluctuate independently of each other; therefore class distributions (proportion of illegitimate examples to legitimate examples) will change over time. Multiple types of fraud can happen at around the same time. Each type can have a regular, occasional, seasonal or once off temporal characteristic. Legal characteristics/behaviour can change over time.

There are basically three types of approaches to solve the problem of detecting fraud in the financial systems. First, Labelled training data can be processed by single supervised algorithms. Labelled data has both activity information and information about its legitimacy i.e activity is legitimate or illegitimate. Supervised algorithms only can be deployed on labelled data. A better suggestion is to employ hybrids such as multiple supervised algorithms , or both supervised and unsupervised algorithms to compare outputs on evaluation data. Second, all known legal claims/transactions sequences should be used processed by semi-supervised algorithms to detect significant anomalies from consistent normal behaviour. Third, single or multiple unsupervised algorithms can be used to process data combining training data with evaluation data to output suspicion scores, rules and/or visual anomalies on evaluation data.

2.1 Supervised Approaches on Labelled Data

Predictive supervised algorithms examine all previous labelled claims/transactions to mathematically adapt a model how a standard fraudulent transaction looks like by assigning a cost function ([34]). Multiple layer Neural networks and support vector machines (SVMs) very popular and have been applied enormously.

A three-layer forward-feeded Radial Basis Function (RBF) neural network has been proposed by [18]. This neural network takes only two training passes to produce a fraud score in every two hours for new credit card transactions. Barse et. al. [5] used a multi-layer neural network with exponential trace memory to handle temporal dependencies in synthetic video-on-demand log data. Syeda et. al. [37] proposed fuzzy neural networks on parallel machines to speed up rule production for customer-specific credit card fraud detection.

The neural network and Bayesian network comparison study of Maes et. al. [24] shows the uses the STAGE algorithm for Bayesian networks and backpropagation algorithm for neural networks in credit transactional fraud detection. Comparative results show that bayesian networks were more accurate and much faster to train, but bayesian networks were slower when applied to new instances.

Ezawa and Norton [14] developed Bayesian network models in four stages with two parameters. They argued that regression, nearest-neighbour, and neural networks are too slow process to make decision.

Statistical modelling has been tremendously utilized and used to detect fraud events. Foster and Stine [16] used least squares regression and stepwise selection of predictors to show that standard statistical methods are also attractive and competitive. Their version of fully automatic stepwise regression has three useful modifications: firstly, it organises calculations to accommodate interactions; secondly, it exploits modern decision-theoretic criteria to choose predictors; thirdly, it estimates p-values to handle sparse data and a binary response before calibrating regression predictions. If cost of false negative is much higher than a false positive, their regression model obtained significantly lesser misclassification costs than C4.5 for telecommunications bankruptcy prediction.

2.2 Hybrid Approaches on Labelled Data

Different popular supervised algorithms such as neural networks, Bayesian networks, and decision trees have been applied in a sequential fashion to improve results.

Chan et. al. [10] utilises naive Bayes, C4.5, CART, and RIPPER as base classifiers and stacking to combine them. They also examine bridging incompatible data sets

from different companies and the pruning of base classifiers. The results indicate high cost savings and better efficiency on credit card transactions.

Phua et. al. [30] proposed backpropagation neural networks, naive Bayes, and C4.5 as base classifiers on data partitions derived from minority oversampling with replacement. Its originality lies in the use of a single meta-classifier (stacking) to choose the best base classifiers, and then combine these base classifiers' predictions (bagging) to produce the best cost savings on automobile insurance claims.

He et. al. [19] proposed genetic algorithms to determine optimal weights of the attributes, followed by k-nearest neighbour algorithm to classify the general practitioner data. They claim significantly better results than without feature weights and when compared to CBR (Case Based Reasoning).

2.3 Unsupervised Approaches on Unlabelled Data

Link analysis and graph mining are hot research topics in anti-terrorism, law enforcement, and other security areas. But these techniques seem to be relatively under-rated in fraud detection research.

A white paper ([26]) describes how the emergent algorithm group is used to form groups of tightly connected data and how it led to the capture of an elusive fraud group by visually analysing twelve months worth of insurance claims.

There is a brief application description of a visual telecommunications fraud detection system ([12]) which flexibly encodes data using color, position, size and other visual characteristics with multiple different views and levels. The intuition is to combine human detection with machine computation.

Cortes et. al. [11] examines temporal evolution of large dynamic graphs' for

telecommunications fraud detection. Each graph is made up of subgraphs called Communities Of Interest (COI). To overcome instability of using just the current graph, and storage and weight problems of using all graphs at all time steps; the authors used the exponential weighted average approach to update subgraphs daily. By linking mobile phone accounts using call quantity and durations to form COIs, the authors confirm two distinctive characteristics of frauds. First, fraudulent phone accounts are linked - frauds call each other or the same phone numbers. Second, fraudulent call behaviour from flagged frauds are reflected in some new phone accounts - frauds retaliate with application fraud/identity crime after being detected. They state their contribution to dynamic graph research in the areas of scale, speed, dynamic updating, condensed representation of the graph, and measure direct interaction between nodes.

Some forms of unsupervised neural networks have been applied. Dorronsoro et al [13] creates a non-linear discriminant analysis algorithm which do not need labels. It minimises the ratio of the determinants of the within and between class variances of weight projections. There is no history on each credit card account's past transactions.So all transactions have to be segregated into different geographical locations. The authors explained that the installed detection system has low false positive rates, high cost savings, and high computational efficiency.

Burge and ShaweTaylor [9] used a recurrent neural network to form short-term and long-term statistical account behaviour profiles. Hellinger distance is used to compare the two probability distributions and give a suspicion score on telecommunications toll tickets.

In addition to cluster analysis , unsupervised approaches such as outlier detection,

spike detection, and other forms of scoring have been applied. Yamanishi et. al. [41] demonstrated the unsupervised SmartSifter algorithm which can handle both categorical and continuous variables, and detect statistical outliers using Hellinger distance, on medical insurance data.

Bolton and Hand [6] recommended Peer Group Analysis to monitor inter-account behaviour over time. It compares the cumulative mean weekly amount between a target account and other similar accounts (peer group) at subsequent time points. The distance metric/suspicion score is a t-statistic which determines the standardised distance from the centroid of the peer group. The time window to calculate peer group is thirteen weeks and future time window is four weeks on credit card accounts. They also suggest Break Point Analysis to monitor intra-account behaviour over time. It detects rapid spending or sharp increases in weekly spending within a single account. Accounts are ranked by the t-test. The fixed-length moving transaction window contains twenty-four transactions: first twenty for training and next four for evaluation on credit card accounts.

Brockett et. al. [8] recommends Principal Component Analysis of RIDIT scores for rank-ordered categorical attributes on automobile insurance data.

Hollmen and Tresp [20] present an experimental real-time fraud detection system based on a Hidden Markov Model (HMM).

2.4 Other Methods

In the literature of social networks, detection of community in them includes the following techniques below. The algorithms proceed to find following things.

Clique: subgroups whose members are all “friends” (i.e. connected with an edge)

to each other. Clique finding algorithms are used [32]. N-clique with two variants : maximal sub-graphs such that the distance of each pair from its vertices is not greater than n [22].

K-plex: maximal subgraph in which each vertex is adjacent to all other vertices of the subgraph except at most k of them [35].

LS-set (weak community) : subgraph such that the internal degree is greater than the external degrees [15].

Lambda set: subgraph where each pair of vertices has a greater edge connectivity than any pair formed by one vertex of the subgraph and one outside the subgraph [36].

Detecting community approaches also include techniques based on either a fitness measure or a quality measure, communities determined by means of modularity-based algorithms, clusters: communities derived using well-known clustering methods [33], traditional methods based on clustering like k-means [23] and others applications [4], divisive algorithms mainly based on hierarchical clustering [29], modularity-based algorithms [25], [38], [1] and other similar algorithms, spectral algorithms [2], dynamic algorithms [40], [21] and other similar algorithms, statistical inference-based methods [3], multi-resolution methods [31], centrality based techniques built around the Newman algorithm [25], local methods around the k-Clique percolation method [17], modularity optimization methods around the Newman algorithm [28], spectral partitioning methods around Simon's algorithm [58] and lastly physics-based methods inspired by Potts law [40].

Chapter 3

Methodology

In this chapter, we have discussed about how a question and answer forum can be represented and proposed a basic scoring procedure. We have shown how to use an extension of graph modularity. We have also proposed a technique to extract possible fraudulent information using sub-graphs and superimposed graphs. We have introduced a measure to quantify collaboration of two users in fraudulent activities.

3.1 Representation of Online Communities

Graph is one of the main models to study relationships. Structural properties of social networks can be measured by representing individuals and their relationships as graphs and computing the centrality of their nodes. Similarly, once a social graph is available, groups of *strongly connected* individuals(communities, in our case fraud voting groups) can be identified using *clustering algorithms*.

In these online communities, *question*, *answer* and *user* are three basic related entities. Users or actors actually represent a virtual accounts created in the communities. We represent these entities by *nodes* (or *vertices*) in a graph and the relations among these entities by connected lines (edges) between the nodes.

According to Wasserman et. al. [39], social networks contain information along at least three different dimensions: a *structural dimension* corresponding to the social graph, e.g. actors/users and their relationships, a *compositional dimension* describing the actors, e.g. their personal information, and an *affiliation dimension* indicating group memberships. In case of question and answer communities, they do not contain information along affiliation dimension. We used *attributed graphs* to represent structural and compositional information of question and answer communities.

Definition 3.1.1. (*Relational Attributed Graph*) Given a set of nodes, N and a set of labels, L and a set of attributes, A, an attributed graph is a quadruple, $G = (V, E, L, A)$ where $V \subseteq N$, $f : E \rightarrow S$ and $l : E \rightarrow L$. Each edge $e \in E$ in the graph has an associated label $l(e)$ and value $f(e)$. Each node $v \in V$ in the graph has associated attributes from A. Each $a \in A$ has a mapping function $g : A \rightarrow W(a)$, where $W(a)$ is the set of values that a can take on.

Figure 3.1 shows an attributed graph with $L = \{black, blue\}$, $S = \{owns\} \cup R$ and $A = \{type\}$. Here, R is set of real numbers and '*owns*' is a nominal value to represent ownership relation. Nominal values are those do not have quantitative meaning, e.g. if we represent '*owns*' as 1, it does not bear any special meaning. So each edge can have only value from S. Each nodes in the graph has an attribute 'type' can take on three values ($W(type) = \{triangle, rectangle, circle\}$) namely 'triangle', 'rectangle' and 'circle'. In Figure 3.1, we have used different shape of nodes to indicate different types, since this way brings better visual understanding.

Users can give an up-vote to a question or an answer if they like the question or the answer. Users can also give a down-vote if they dislike the answers or the questions.

In the Figure 3.1, we have shown three types of nodes by using circle, triangle and rectangle to represent user, question and answer respectively. Two types of edges are

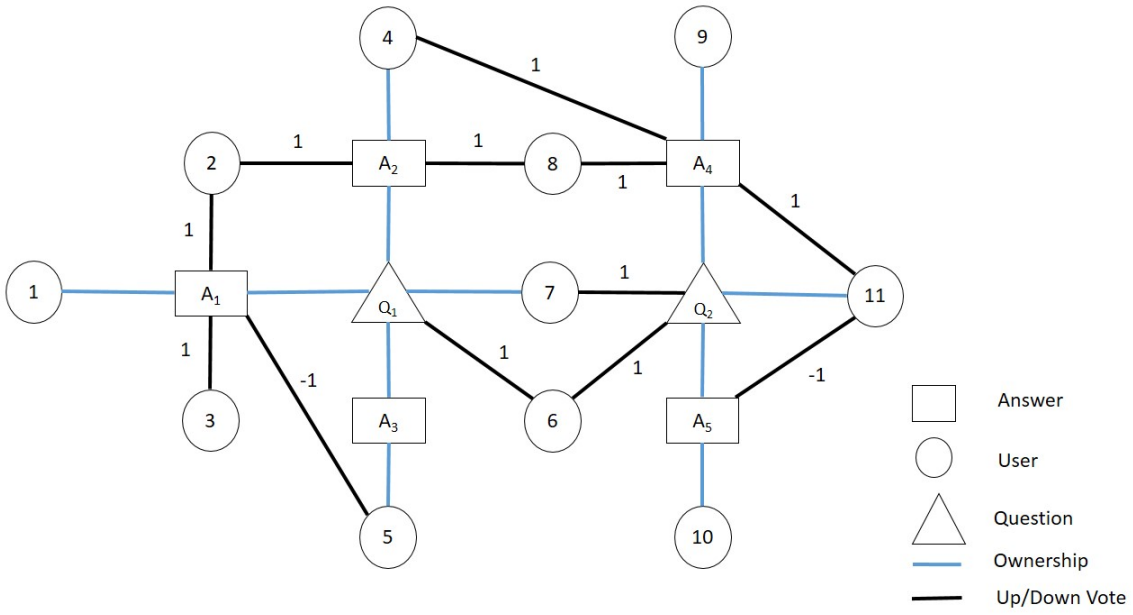


Figure 3.1: Representation of online communities in an attributed graph model. Circles, rectangles and triangles are used to represent users, answers and questions. Up-vote and down-vote points are assumed to be 1 and -1 respectively.

labeled by blue and black colors to represent *ownership* relation and points credited (*up-vote*) or points penalized by a user (*down-vote*) respectively. Ownership relation can exist between a user and a question or an answer. Black edges have weights equal to up-vote points or down-vote points. Up-votes are assumed to be always positive and down-votes to be negative real numbers.

All users can give an up-vote or down-vote to an answer but the questioner cannot give an up-vote or down-vote to his/her own question. So no black edge can occur between user node and question node. The questioner also cannot give an up-vote or a down-vote to his own answer to this question. So no multi-edge or loop can occur.

We have used blue edges between user node and answer or question node to mean that the answer belongs to that question (ownership). So there must be only one

blue edge happen between a question node and user node. Exactly two blue edges can be incident to an answer node, one edge represents a user owns the answer and other represents that a question owns the answer. In the Figure 3.1, 1, 2, 3, ..., 11 are user nodes (circles), Q_1, Q_2 are question nodes (triangles) and $A_1, A_2, ..., A_5$ are answer nodes(rectangles). For examples, the question Q_1 posted by user 7 has answers A_1, A_2, A_3 given by users $1, 4, 5$ respectively. The answer A_1 is up-voted by user 3 and down-voted by user 5 as positive black edges indicate up-votes and negative black edges indicate down-votes.

3.2 Scoring System

In this section, we have shown a procedure to calculate scores of the users to show that our proposed graph model is sufficient to calculate scores. We have also associated scores to all type of nodes. The score obtained by a user is popularly known as rating point in the communities. The score of an question measures how good or useful the question is to the members of the communities. The score of an answer measures the correctness or the acceptability of the answer to users of the communities as well as to the questioner. We denote *QScore*, *AScore* and *UScore* as score obtained by a question, an answer of a question and a user/actor at a certain point of time respectively.

Let $E(x)$ be the set of black edges incident to the question node x. Then *QScore* of a question can be defined as the Equation 3.2.1:

$$QScore(x) = \sum_{e \in E(x)} w(e) \tag{3.2.1}$$

Here, $w(e)$ denotes weight of the edge e. For examples, *QScore*s of Q_1 and Q_2 are

respectively 1 and 2 in Figure 3.1.

Let $F(x)$ be the set of black edges incident to the answer node x. Then *AScore* of an answer is defined as the Equation 3.2.2.

$$AScore(x) = \sum_{e \in F(x)} w(e) \tag{3.2.2}$$

For examples, *AScore*s of A_1, A_2, A_3, A_4, A_5 are 2, 2, 0, 3, -1 respectively in the Figure 3.1.

Let $A(x)$ and $Q(x)$ be the set of answer nodes and question nodes incident to a user node x ($A(x)$ and $Q(x)$ contains answers given by x and questions posted by x respectively) respectively, then *UScore* can be defined by the Equation 3.2.3.

$$UScore(x) = \sum_{n \in Q(x)} QScore(n) + \sum_{m \in A(x)} AScore(m) \tag{3.2.3}$$

For examples, *UScore*s of 1, 7 and 9 are 2, 1 and 3 respectively in Figure 3.1. A very simple scoring procedure is proposed in the Algorithm 1. The algorithm is very simple, self-descriptive and no explanation is necessary. This algorithm conforms that attributed graph model is valid and sufficient to calculate rating points.

3.3 Answer Sub-graph and Superimposed Graph

Clustering is a process of assigning different nodes to different groups. It is an important operation whose best known application is the discovery of communities in social networks. Graph clustering and community detection have traditionally focused on graphs without attributes, with the notable exception of edge weights. However, these models only provide a partial representation of real social systems, that are thus often described using node attributes, representing features of the actors, and edge

Algorithm 1 Algorithm for scoring system

Input: Attributed graph $G = (V, E, L, A)$; where $V = Q \cup A \cup U$; Q, A and U are the sets of question, answer and user nodes, L is the set of edge labels; A is the set of node attributes;

initialize the score of every nodes to 0;

foreach *Question node q in Q* **do**
| calculate $QScore(q)$ using Equation 3.2.1
end
foreach *Answer node a in A* **do**
| calculate $AScore(a)$ using Equation 3.2.2
end
foreach *User node u in U* **do**
| calculate $UScore(u)$ using Equation 3.2.3
end

attributes,representing different kinds of relationships among them. We have presented our question and answering communities by attributed graph in the section 3.1.

Now, we introduce the notion of answer sub-graph of an answer. An answer sub-graph of an answer is a sub-graph containing that answer node, the black edges incident to that answer node and user nodes incident to these black edges. In other words, the answer sub-graph of an answer contains only that answer node, the user nodes who have up-voted/down-voted that answer and the black edges in between them. In the Figure 3.2 (a), (b) and (c) are answer sub-graphs formed by answers A_1, A_2, A_3 respectively.

Definition 3.3.1. *(Answer Sub-Graph)* Given an attributed graph, $G = (V, E, L, A)$, a sub-graph formed by an answer node X is $G(X) = (V_s, E_s, B_s, A_s)$, where $V_s \subseteq V$ and $V_s = A \cup U$, $U = \{$ User node $u : u$ is incident to $A\}, E_s \subseteq E$, and
E $= \{e : \exists u \in U$ and $(A, u) = e\}$

Definition 3.3.2. *(Flattening and Superimposed Graph)* Given a set of N answer sub-graph formed by user node i $G_i = (V_i, E_i, L, A)$, $i = 1, 2, ..., N$. N is the total number of answers given by user i. Fattening of some answer sub-graphs is a forming

of a new graph such that all common nodes are superimposed on one another to form new common nodes and common edges are superimposed on one another to form new common edges with new weights. If some nodes or edges are not common, they are also added to the graph. Superimposed graph formed by N answer sub-graph $G_i = (V_i, E_i, L, S)$, i = 1, 2, ..., N is a graph $G = (V, E, L, A)$, where $V = \cup V_i$, $E = \cup E_i$, $B = \cup V_i$, $\forall e \in E\ w(e) = f(w_1(e), w_2(e), ..., w_N(e))$ and $f : R \rightarrow R$. Here, $w_i(e)$ is weight of edge e in G_i.

In the Figure 3.2 (d), a superimposed graph is shown by flattening answer sub-graphs of 3.2 (a), 3.2 (b) and 3.2 (c). An important thing to notice that some edges are more thicker than the others. But the new weights of black thick edges are calculated with the formula of 3.3.1.

Let $w_i(u, v)$ be the weights of edges incident to node u and v in answer sub-graphs formed by user node i and new weight of the edges of superimposed graph is $w_{\text{sup}}(u, v)$.

$$w_{\text{sup}}(u, v) = \sum_{i}^{N} w_i(u, v) \tag{3.3.1}$$

Here, N is the total number of all answer sub-graph formed by user node i. In the Figure 3.2, new weights of the black edges are shown as label. New weights are merely the summation of weights in the answer sub-graphs. In the Figure 3.2, edges are thickened as to stress conceptual clarity, otherwise thickening bears no meaning while new edge weights are given.

An intuitive and clear indication from the Figure 3.2 that together B, X, Y and Z nodes can be clustered as a group. Since a greater portion of B's *UScore* comes from X, Y and Z, we can conclude there is a big contribution to B's *UScore*(rating point) from X, Y, Z. If we can find that, to X, Y and Z's *UScore* there are big contributions from B, Y, Z; B, X, Z; B, Y, X, then there is good possibility that B, X, Y and Z have formed a illegal voting group.

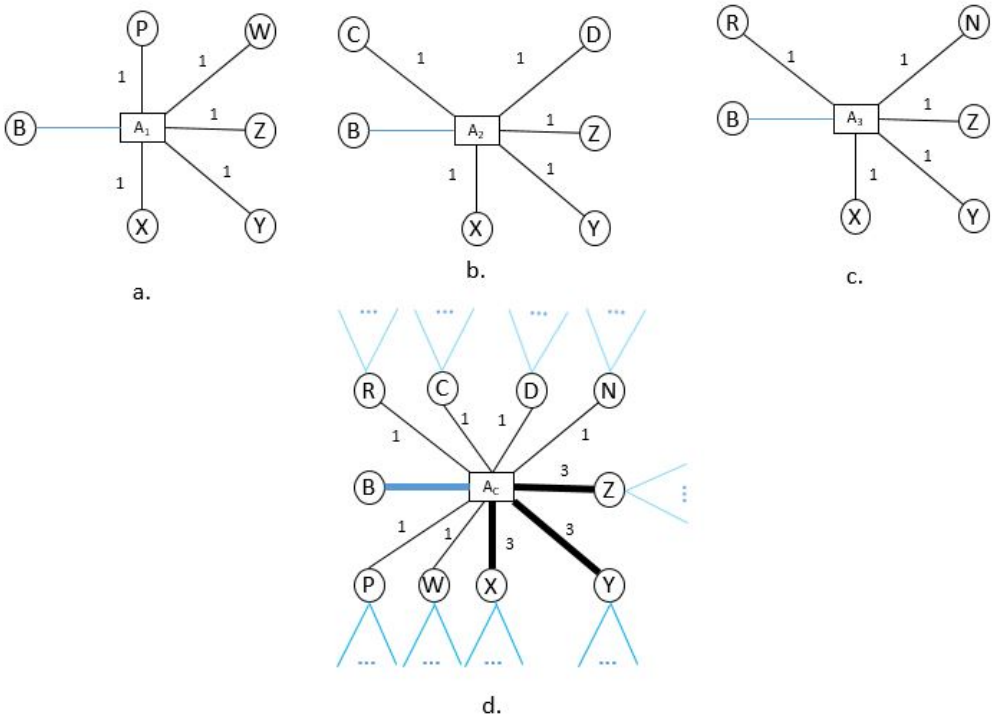


Figure 3.2: Answer sub-graph formed by (a) B and A_1 (b) B and A_2 (c) B and A_3 (d) superimposed graph of A_1. Notice the thick-edges, new weight is calculated by graph-flattening formula 3.3.1. Ellipsis(...) after the blue edges hint that those nodes can also own answers and questions that are not part of superimposed graph. A_1, A_2, A_3 constitute new node A_c. User node B and incident blue edges are shown just to stress good understanding, they are not part of their corresponding graphs.

In the later sections, we have proposed a extended modularity measure and a proximity measure to form a threshold graph. Then, finding clusters, maximal connected components or strongly connected components can find probable voting group. The definitions and concerned notions will be explained.

3.4 Graph Modularity and Extended modularity

Modularity is a measure of how well the nodes in a graph can be separated into dense and independent components [27]. Figure 3.3 shows four graphs with their nodes assigned into two communities (black and white) and the modularities resulting from these assignments. In these examples, it clearly appears how the assignments putting together highly interconnected nodes and separating groups of nodes with only a few connections between them get a higher value of modularity. The modularity thus expressed as the Equation 3.4.1

$$Q = \frac{1}{2m} \sum_{\text{ij}} (a_{\text{ij}} - \frac{k_i k_j}{2m}) \delta(\gamma_i, \gamma_j) \tag{3.4.1}$$

Here, k_i and k_j are degree of the nodes i and j respectively.

where $\delta(\gamma_i, \gamma_j)$ is the Kronecker delta which returns 1 when nodes i and j belong to the same cluster, 0 otherwise. Therefore, the sum is computed only for those pairs of nodes that are inside the same cluster. For each of these pairs, the presence of an edge between them improves the quality of the assignment: a_{ij} equals 1 when there is an edge between i and j, 0 otherwise. As we are dividing everything by m (the number of edges in the graph), edges between nodes belonging to different clusters negatively affect modularity because they are not considered in the numerator (as $\delta(\gamma_i, \gamma_j) = 0$), but are counted in the denominator (m). Finally, the formula 3.4.1 considers the fact that two nodes with high degree would be more likely to end up in the same cluster by chance.

In practice, modularity is a good evaluation formula to measure how well a clustering algorithm has formed its clusters. But in the literature of social networks, modularity is also extended by researchers to find clusters. We also proceed to find

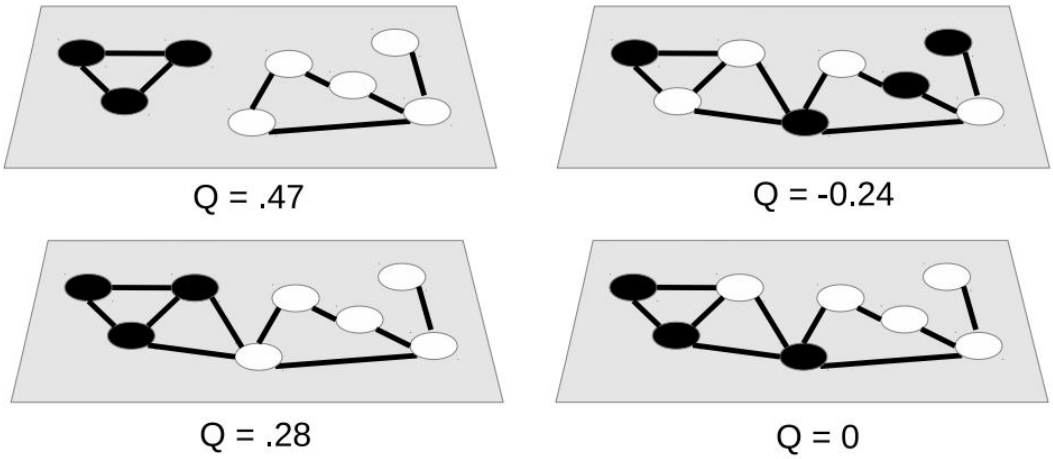


Figure 3.3: Modularity of four graph clusterings: nodes in each graph are assigned to two clusters (black and white); the modularity of each assignment is shown under the graph. Figure courtesy : [7]

voting groups by extending moduarity to find clusters rather than evaluating clusters.

3.4.1 Extension of Modularity

We have extended the idea of modularity such a way to adopt it to superimposed answer graphs. The users who contributes greater to the extended modularity of superimposed answer graph will have a Kronecker delta $\delta(\gamma_i, \gamma_j)$ closer to 1 and for the user which contributes smaller to the extended modularity of superimposed answer graph will have a Kronecker delta closer to 0. The Equation 3.4.2 we have shown the changes made to Kronecker delta. The smaller value of extended Kronecker delta indicates the node would be classified this group with smaller probability. The reverse argument holds when Kronecker delta is greater.

Let m be the total number of (black) edges in the superimposed answer graph, S be the sum of all edges E be the set of all edges in superimposed answer graph.

$$\delta(\gamma_i, \gamma_j) = \frac{w(e)}{m} = \delta(e) \tag{3.4.2}$$

The extended modularity of a superimposed graph of user node u can be expressed as the Equation 3.4.3.

$$Q(u) = \sum_{e \in E} \left(\frac{w(e)}{\sum_{e \in E} w(e)} - \frac{1}{m}\right) \frac{w(e)}{m} \tag{3.4.3}$$

The equation 3.4.3 has an proportional relation to activity level of the user. For example, user A and B's superimposed answer graph has same number of nodes i.e. they have relation with same number of users. If we say, user A has been up-voted 1000 times and user B has been up-voted 500 times then A will have greater extended modularity. This conforms with the fact that a relatively new user with same number of connected people has a less chance to be involved in a voting group. But an relatively aged user with same number of connected people has greater chance to be involved in a voting group.

In the Figure 3.4 (a) and (b) we have compared the measure of extended modularity. Graph of Figure 3.4 (a) has a more dense component than in graph of Figure 3.4 (b). Graph of Figure 3.4 (a) and Figure 3.4 (b) has modularity are 0.042 and 0.014 respectively. We have noticed that graph of 3.4 (b) has black edges with a similar weights. So we can not bring any good possibility of fraudness to the scenario. In the Equation 3.4.3, we adapted Kronecker delta to superimposed answer graph only considering the black edges. In both Figures, 3.4 (a) and 3.4 (b) have m = 6 and total weight sum = 12. So the graphs of both the Figures has same amount of history. But since Figure 3.4 (a) has greater activity with smaller number of nodes than that of Figure 3.4 (b).

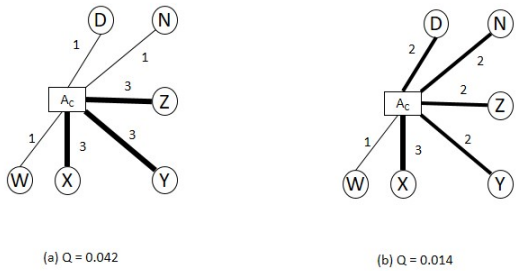


Figure 3.4: Extended modularity of superimposed answer graphs shown on graph : superimposed answer graph in (a) has a higher extended modularity than (b).

The Equation 3.4.3 can have smallest value negative infinity ($-\infty$) and 0 if a user has no activity and some activity is present. The Figure 3.5 shows the cases when extended modularity can have lowest value.

3.5 Proximity Measure between User Node

Now we have a way to calculate extended modularity of user nodes from its associated superimposed answer graph. We have studied the system using a proximity measure between the user nodes (actors). This proximity measure quantifies the possibilities of participating in a voting group of two users. Proximity between two user A, B is defined as the Equation 3.5.1:

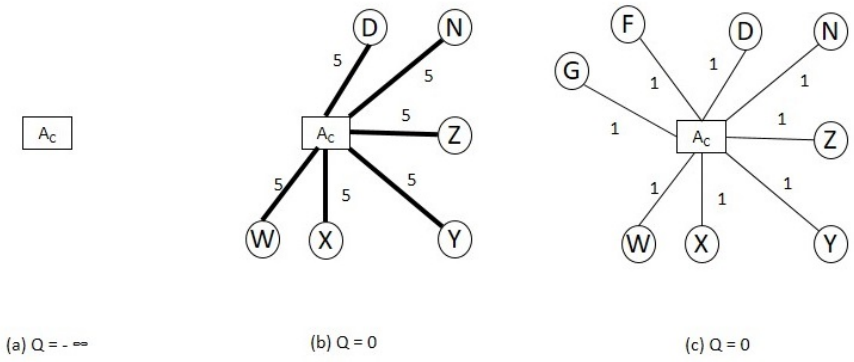


Figure 3.5: Extended modularity value of superimposed answer graphs shown on graph :(a) the user has no history $Q = -\infty$ (b) and (c) the users have only one connection with other users, $Q = 0$

$$P(A,B) = Q(A)\frac{\sum_{e\in SS} w(e)}{\sum_{e\in S} w(e)} + Q(B)\frac{\sum_{e\in RR} w(e)}{\sum_{e\in R} w(e)} \tag{3.5.1}$$

Here, R is the set of black edges with positive weights(up-vote edges) that are incident to A. RR is the set of black edges with positive weights(up-vote edges) incident to both A and B's superimposed answer graph's superimposed answer node. S is the set of black edges with positive weights(up-vote edges) that are incident to B. SS is the set of black edges with positive weights(up-vote edges) going from B to A's superimposed answer graph's superimposed answer node. The first sum term ratio means the fraction of up-votes contributed by B to A's superimposed graph and second term sum ration refers to the fraction of up-votes contributed by A to B's superimposed answer graph. Higher value of $P(A,B)$ indicates higher chances of A, B are involved in the same voting group.

The lowest value of the extended modularity between two user nodes can be 0. $P(A, A)$, $\forall A \in U$ (set of user nodes) is 0 since a user cannot vote to him/herself. This is an exception with normal similarity measures since similarity between the same variable is maximum. Another exception is that $P(A, B)$ is not bounded to a highest value. In first case in Figure 3.5 (a), when one of the users has $Q = -\infty$, there is no collaboration between these two, so $P(A, B)$ must be 0. In second case in Figure 3.5 (b) and (c), when both the users has $Q = 0$, there is no collaboration between these two, so $P(A, B)$ must be 0.

Since extended modularity is somewhat increasing function of activity level of the users. Here, activity level means how much active (history) is the user on the system. As percentage of vote cannot exactly tell much about forming a group. For example, a user gave only 2 votes (probably new or inactive user) to the answers and another user gave 10000 votes to the answers. So same percentage amount of votes involved in the vote sharing does not have same effect. For example, both user shared 50% (first user : 1 vote, second user : 5000 votes) votes to another user C. Practical experience shows that, possibility of forming vote group with user C of second user is greater that first user. $P(A, B)$ can be also 0 when there is no vote sharing between A and B.

The Equation 3.5.1 measures the amount of collaboration of two users in a probable voting group. 'Plus' sign of the Equation 3.5.1 has an explantion. Sharp readers can argue why 'plus' instead 'multiplication'. The reason is that say, user A has an sock puppet (fake account) B to increase rating point of A, so only B has voting collaboration towards A not vice versa. If 'multiplication' is used instead then whole term would be 0. But that is not what we want.

	B	X	Y	Z	D	N
B	0	0.5	0.45	0.45	0.1	0.12
X	0.5	0	0.52	0.65	0.11	0.13
Y	0.45	0.52	0	0.65	0.20	0.25
Z	0.51	0.55	0.65	0	0.26	0.29
D	0	0	0	0	0	0.76
N	0.19	0.09	0.2	0.1	0.76	0

Table 3.1: Resulting proximity between every pair of user nodes B, X, Y, Z, D, N.

3.6 Threshold Graph and Clustering

A concept that is closely related to the clustering algorithms based on graph theory is threshold graph. Threshold graphs can reveal the clusters formed by users in the social networks. Definition of threshold graph is as below.

Definition 3.6.1. *(Threshold Graph)* Let a be a similarity (dissimilarity) measure and ω (threshold) be real number. A threshold graph $G(\omega)$ with N nodes contain an edge between two nodes i and j if he similarity(dissimilarity) between i and j is greater (less) that or equal to $\omega, i, j = 1, 2, 3, ..., N$. We may write $(v_i, v_j) \in G(\omega)$ if $a(i, j) \geq (\leq)\omega$, $i, j = 1, 2, 3, ..., N$, if $a(i, j)$ is a similarity (dissimilarity) measure.

Now we can form a proximity matrix using our proposed proximity measure between all pair of users (user nodes) and save it in a matrix to be later used by a clustering algorithm (divisive, aglomerative algorithms).

Table 3.1 contains information between every pair of users exist in a community. The corresponding threshold graph is shown in Figure 3.6 . We notice that every primary diagonal element of the matrix of Table 3.1 is 0. This conforms with the restriction that a user cannot vote him/herself. Some entries are also 0 since no vote sharing between the users. The matrix is symmetric since threshold graph is an undirected graph. The (i, j) entry is equal to (j, i) entry.

Figure 3.6 shows the corresponding threshold graph of Table 3.1 using threshold

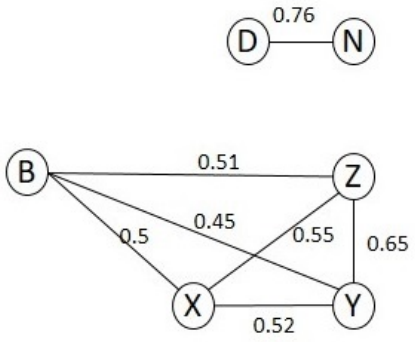


Figure 3.6: The corresponding threshold graph of example of Table 3.1 with threshold (a) 0.30 .

value 0.30 . The figure clearly reveals visually probable voting groups (B, X, Y, Z) and (D, N) if threshold has good relevance with currently available data.

A good choice of the threshold dictates the finding of good and realistic clustering. As the system passes time, threshold value must be updated to find reasonable voting group. Because the proximity value is increasing function of activity level. Machine learning approaches (e.g. regression) can be adapted to system to predict good threshold value. Now we can find the connected subgraph or more rigorously completely connected subgraph (clique) to find vote group. Thus the user nodes of the same completely connected subgraph are more likely to be involved in a vote group. These users should be closely monitored. A choice of threshold is upto the system administrator or adaptive. How much collaboration of up-votes are fraudulent to system to system e.g. one system can set 10% mutual up-votes as fraud another system can set 30% up-votes as fraud.

Definition 3.6.2. *(Maximal Connected Components)* Let $G = (V, E)$ be an undirected graph, connected component of G is a set of edges and nodes such that for

every pair of nodes u, v there exists at least one path (alternating non-repeated sequence of nodes and edges) between them. A maximal connected component of G is a connected component of G that is not contained in other connected components.

Definition 3.6.3. *(Maximal Clique)* A clique, C, is an undirected graph $G = (V, E)$ is a subset of the nodes, $C \subseteq V$ such that every two distinct pair of nodes are adjacent. This is equivalent to the condition that the subgraph of G induced by C is complete. A maximal clique is a clique that is not contained in other cliques.

The Algorithm 2 shows the basic steps to find probable voting group. Well known maximal connected component finding algorithms (e.g. DFS algorithm) to find voting groups. As finding all the maximal cliques in a graph in exponential in time, finding maximal connected component is a better approach. So to find a group that all members collaborate to each other, we can pass these connected components to clique finding algorithms as input.

Algorithm 2 Algorithm for Finding Voting Groups

Input: Attributed graph $G = (V, E, L, A)$; $V = Q \cup A \cup U$; Q, A and U are the sets of question, answer and user nodes, L is the set of edge labels, A is the set of node attributes;
A threshold ω;
An input empty undirected graph $G_{un} = (V, E) V_{un} = \emptyset, E_{un} = \emptyset$;

Output: A set of clusters (voting groups)

foreach $u \in U$ **do**
  form superimposed answer-graph of u from G;
  calculate extended modularity, $Q(u)$ using the Equation 3.4.3;
end
foreach $u, v \in U$ **do**
  calculate proximity $P(u, v)$ using the Equation 3.5.1;
  if $u \notin V_{un}$ **then** $V_{un} \cup u$;
  if $v \notin V_{un}$ **then** $V_{un} \cup v$;
  if $P(u, v) \geq \omega$ **then** $E_{un} \cup (u, v)$ with $\omega(u, v) = P(u, v)$;
end
find all maximal connected components in graph G_{un};
add the user nodes in the same connected component to same voting group;

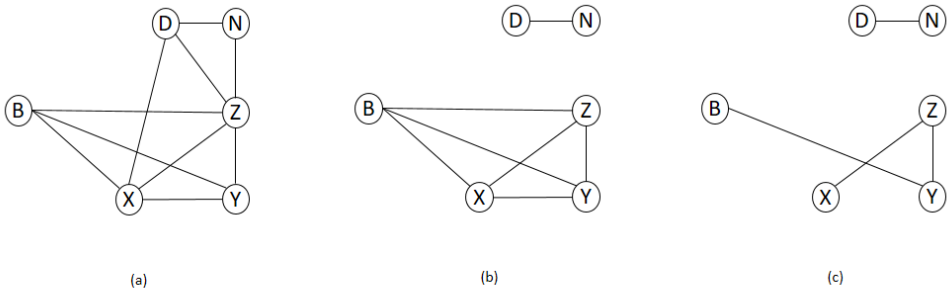


Figure 3.7: (b) Good choice of threshold uncovers voting group (B, X, Y, Z) and (D, N) (a) (c) bad choice of threshold puts all nodes to same group or removes other potential member of the groups respectively.

The effect of bad choice of threshold is shown in the Figure 3.7. A smaller threshold relative to the existing system's actual threshold tends to find group with members actually not in the groups. A greater threshold relative to the existing system's actual threshold finds group with very strong fraudness in them. A lower sharer of these group is left out of the group.

Chapter 4

Experimental Results

In this section, we have shown step by step procedures to detect fraud groups in the online communities. We have also shown experimental results and graphs to analyze the datasets.

4.1 Experimental Dataset

We have collected datasets from online application and also synthesized a dataset. In this section, we have briefly described the datasets. We have used two different datasets :

1. Synthesized data
2. Durbin Community Application data

Synthesized Data: We have generated synthetic data where we have intentionally put some fraudulent activities and tested our algorithm with this synthetic data. This synthetic data is also available in public domain in the following link: `https://www.github.com/mostafiz93/synthetic.data`

The dataset contains:

- 30 user nodes
- 67 posts (answer nodes)
- 145 votes

Durbin Community Data: The dataset is collected from a online community platform of Durbin Labs. The dataset contains following contents:

- 10544 user nodes
- 1392 posts (answer nodes)
- 11914 votes

With due permissions from Durbin Labs, the proprietorship of the data has been removed from this amount of data, we have made available the data in public domain (`https://www.github.com/mostafiz93/Thesis/durbin.zip`) so that anyone can experiment our algorithm with this data. The dataset is collected over a period from October 3, 2014 to February 14, 2016.

4.2 Steps to Process Data

In the chapter 3, we have described all related methodologies in elaboration. In this section, we have briefly reviewed basic steps to process data. The steps are listed as below.

1. Calculate extended modularity of every user nodes
2. Calculate proximity collaboration between every pair of user nodes

3. Choosing a good threshold (ω) and represent the nodes in a threshold graph using this threshold (ω)

4. Find maximal connected components (find maximal clique in the maximal connected components)

5. Change threshold to find better result and go to step 3

Step 1: Superimposed graph is formed from dataset for every user nodes. A thing to remember that user, answer and question are all different types of nodes. Extended modularity is calculated for every user nodes using the Equation 3.4.3. Formation of explicit superimposed graph is not needed, only calculated score is necessary.

Step 2: Proximity collaboration score is calculated using the Equation 3.5.1. Necessary condition should be checked to avoid unwanted result for the measure i.e infinity score of extended modularity.

Step 3: In this step, a value of threshold is chosen. A good choice of threshold is important to find actual fraud group. Threshold is a user defined value, how much activity level is said to be fraud is upto user's necessity. We analyzed dataset using different thresholds.

Step 4: Proximity scores are converted to a threshold graph using a threshold and maximal connected components are searched using algorithm e.g. DFS algorithm. To find rigorous and complete group, maximal connected component can be searched for maximal cliques. Maximal connected components find incomplete groups.

Step 5: Fraud groups are closely monitored for actual fraudness. To find better result, threshold can be changed and the procedure can be repeated. A better approach would be adaptive threshold that is learned from historical data.

4.3 Experimental Results of Synthetic Data

We apply the process steps described in section 4.2 on synthetic data. The dataset contains user nodes numbering from 1, 2, 3, ..., 30. Table 4.1 shows calculated extended modularity values using the Equation 3.4.3. A point is to notice some values are 0 or $-\infty$ as we discussed in section 3.4.1. Then proximity collaboration scores between all pair of user nodes are calculated and saved in a matrix. Most of the entries of that matrix are 0 since it is synthetic data with low amount of activity collaboration between most of the pair are smaller. This proximity matrix has been used to form threshold graph with different threshold. We showed results for 3 different threshold 0.05, 0.10 and 0.15. Figure 4.1, Figure 4.2 and Figure 4.3 shows threshold graphs and results are also explained. Results are self-descriptive and easy to understand. If threshold is increased then voting group would disappear. So a good choice of threshold is necessary.

4.4 Experimental Results for Real Data

Durbin Community data contains to many nodes to show in a threshold graph. So we listed the edges present in the data in tables. We worked with different thresholds. Connected components and maximal cliques are shown as summary.

1	0.055600
2	0.055600
3	0.044400
4	0.222200
5	0.406200
6	0.177800
7	0.303600
8	0.111100
9	0.062500
10	0.125000
11	0.062500
12	0.098200
13	0.000000
14	$-\infty$
15	0.000000
16	$-\infty$
17	0.000000
18	$-\infty$
19	0.000000
20	$-\infty$
21	0.000000
22	$-\infty$
23	0.000000
24	$-\infty$
25	0.000000
26	$-\infty$
27	0.000000
28	$-\infty$
29	0.000000
30	$-\infty$

Table 4.1: Table showing extended modularity scores all 30 user nodes

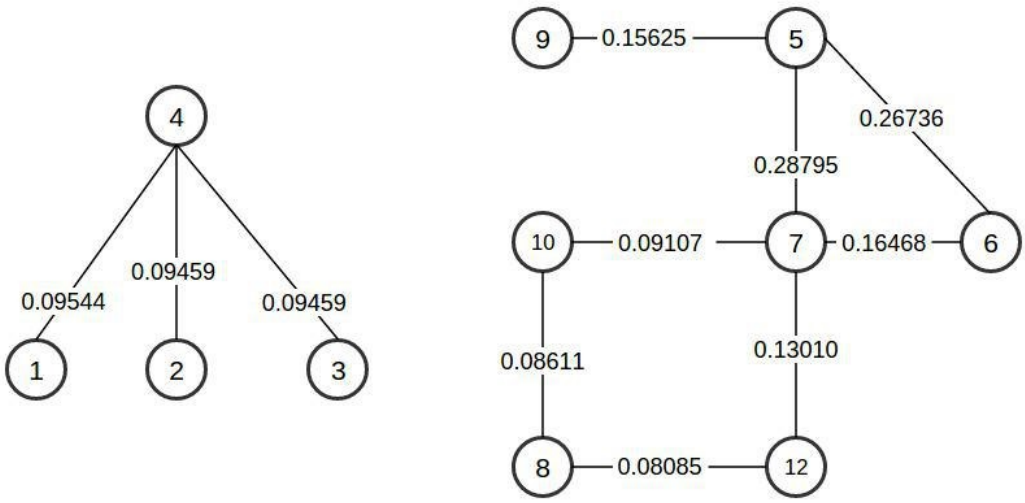


Figure 4.1: Threshold graph with threshold 0.05. Connected components are shown and associate members form incomplete group (1, 2, 3, 4) and (5, 6, 7, 8, 9, 10, 12). Maximal cliques are complete group (8, 9, 10, 12) and (5, 6, 7).

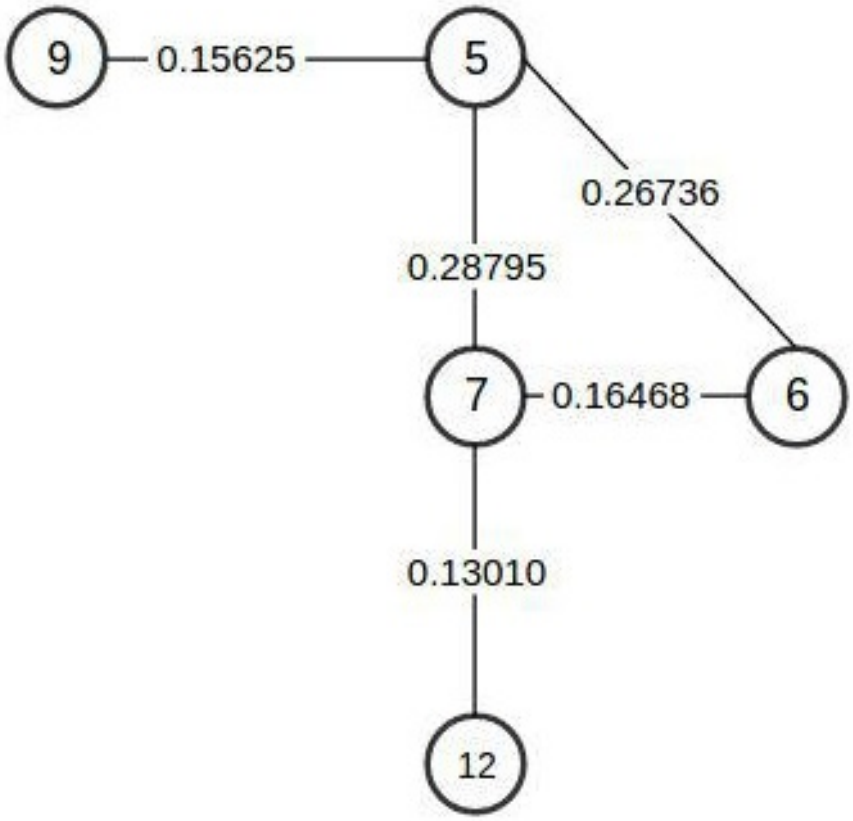


Figure 4.2: Threshold graph with threshold 0.01. Connected components are shown and associate members form incomplete group (5, 6, 7, 9, 12). Maximal cliques are complete group (5, 6, 7). Now real voting group is detected with our synthetic data.

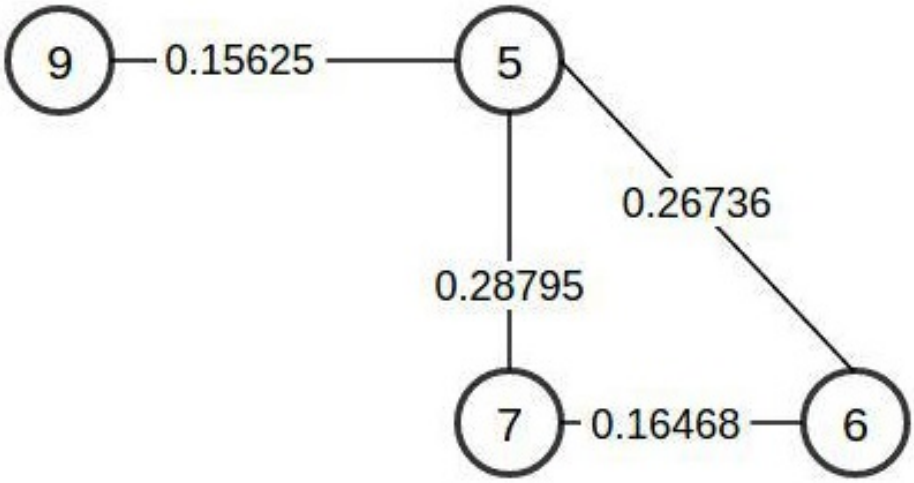


Figure 4.3: Threshold graph with threshold 0.015. Connected components are shown and associate members form incomplete group (5, 6, 7, 9). Maximal cliques are complete group (5, 6, 7). This time also real voting group is detected with our synthetic data.

[4372, 4156] - 0.44230	
[1006, 5701] - 0.88460	[1892, 1965] - 0.44230
[2591, 3964] - 0.44230	[1132, 6231] - 0.35384
[6032, 3616] - 0.44230	[6395, 9315] - 0.44230
[10194, 2055] - 0.35384	[7086, 9507] - 0.37911
[1555, 1323] - 0.44230	[5353, 7099] - 0.88460
[6516, 8360] - 0.44230	[1858, 9467] - 0.44230
[10369, 2403] - 0.88460	[4504, 7171] - 0.88460
[898, 8364] - 0.88460	[9318, 9915] - 0.44230
[3074, 2748] - 0.44230	[9253, 1636] - 0.44230
[8003, 2630] - 0.44230	[3012, 4198] - 0.88460
[8136, 8638] - 0.37911	[1460, 1399] - 0.44230
[2952, 2035] - 0.44230	[7345, 1803] - 0.44230
[9717, 8495] - 0.88460	[3813, 1684] - 0.35384
[8512, 5979] - 0.44230	[2767, 9718] - 0.44230
[9333, 1842] - 0.88460	[8474, 3381] - 0.44230
[8307, 2151] - 0.88460	[9068, 9676] - 0.88460
[4224, 6468] - 0.44230	[3126, 5048] - 0.88460
[4566, 1084] - 0.44230	[4228, 3926] - 0.44230
[7741, 2383] - 0.44230	[9715, 7650] - 0.44230
[7628, 8570] - 0.37911	[5530, 2543] - 0.35384
[3636, 9843] - 0.44230	[46, 3765] - 0.44230
[5083, 5949] - 0.88460	[4358, 2451] - 0.37911
[5242, 4834] - 0.44230	[8926, 5194] - 0.88460
[6900, 7304] - 0.88460	[1104, 4333] - 0.44230
[4530, 3320] - 0.88460	[4290, 5448] - 0.88460
[7908, 745] - 0.44230	[8917, 623] - 0.44230
[2202, 10202] - 0.44230	

Table 4.2: Edges of threshold graph using threshold 0.30. Data is shown in [u, v] - proximity measure format, where u, v are user node number.

[6870, 9480] - 0.88460
[6597, 6805] - 0.88460
[1449, 6477] - 0.88460
[9965, 2077] - 0.88460
[4674, 9958] - 0.88460
[9863, 5057] - 0.88460
[3181, 572] - 0.88460
[8366, 3223] - 0.88460
[5583, 2044] - 0.88460
[4689, 7092] - 0.88460
[10347, 5348] - 0.88460
[10220, 1042] - 0.88460
[2161, 8596] - 0.88460
[2139, 8301] - 0.88460
[9574, 3652] - 0.88460
[9611, 2260] - 0.88460

Table 4.3: Edges of threshold graph using threshold 0.50. Data is shown in [u, v] - proximity measure format, where u, v are user node number.

Chapter 5

Discussion and Conclusion

This article was inspired by the need for measures for quantifying fraudness in social networks (online communities). Based on the concepts of graph modularity and superimposition of graphs, this study develops a collaboration measure to explore fraudulent activities present in the online communities. This collaboration measure can be utilized to quantify fraudness present in the online (question and answer) communities.

In the present literature of detection of fraud group in the online communities, there exists a very few techniques and methods. But its broader category social network contains much amount of techniques and methods to find different groups in the social networks (i.e. methods to find central characters, socio-cultural groups, same interest groups). The proposed measure is capable of uncovering potential fraudulent collaboration between users. Then threshold graph can be exploited to find these probable fraud groups.

The extension of graph modularity is used to define fraudness in an user. This measure stresses on the fact that an user is involved in a fraud group with more probability if more share of his/her votes goes to a small number of users. This

measure also takes into account that the user who contributes smaller amount of votes to another user has a smaller probability to end up in the same voting group. In the same way, the user who contributes greater amount of votes to another user has a greater probability to end up in the same voting group.

The choice of a good threshold is a challenge to find reasonable and real voting groups. This study shows the effect of choosing bad threshold. This study did not show methods to find good threshold values. This task is left for future study. The study takes different threshold to analyze real data and finds that changes in threshold can uncover voting groups. The machine learning (i.e regression, Bayesian predictor with probability distribution) methods could a great method to find good threshold value. Using a cost function that takes into account the cost that could occur by assigning a legitimate user to a illegal voting group and thus this function can be optimized.

The main limitation of this study is that an user can not limit their activities within some of his friend users for a very long time. The more an user confines his/her activity to certain users, the more the chance to be listed as fraud, although a practical group may have not been formed among themselves. This study did not take into account the type (domain) of the online community. Taking domain of community into account (i.e. question-answer forum, economic forum, recreational activity sharing forum etc.) can be helpful to find more voting group more reliably. The same threshold across all parts of the network can be another problem. So different thresholds can be deployed to different parts of the network can produce far better results.

Attributed graph clustering is an active research area, and as such it presents a

number of directions for future works. Early works on attributed graph clustering have focused on finding static communities, which is a preliminary and necessary step to study their evolution(growing or squeezing fraud groups). This study does not also show evolution of voting groups, but it is very easy to understand that it show basic steps to identify frauds in the other types of online communities.

5.1 Future Directions

Clustering of attributed graphs is a recent research area, much of it is related to social networks and communities. There are so many open problems on this topic area. In the article by [7], open problems are classified into two main categories: 1) related to single layer graph are considered (relational information are avoided) 2) related to the combination of structure and attributes.

Overlapping community (social group, fraud group etc) detection, evaluation measures are recent works on single layer graph and can be considered as benchmarks for attributed graph clustering. Measuring the significance of overlapping nodes and interpretation those results are problems for future research.

Defining the analysis context to decide how good a clustering is and to make this context understandable to an analyst without domain knowledge and usable by a domain expert without deep analytical skills are active reasearch area.

In our answer and question forums, we assumed that high percentage of score sharing leads to a probable voting group. But in practical case, high percentage of score sharing can be co-incidental. We also assumed the absolute value of up-votes and down-votes to be equal. But in practical case, they can be of different values. In practical cases, cost function can be associated with marking users as fraud.

While detecting voting group we did not take into account the domain specific knowledge (more attribute values can be taken into consideration). Popularity measures can be introduced to cluster voting group in a very realistic manner. Since popular users can have some fans who tend to like his/her answer. This does not mean to form a voting group within popular person and his/her fans. Social networks analysis can be employed here in this case (detecting popular leader or prestige user in social networks).

Machine learning (i.e regression, Bayesian techniques with Gaussian distribution) and data-mining techniques can be involved to find statistical threshold to cluster users.

We did our experiments with same threshold across the network. But using different threshold across the network or different domain of the network can produce better result.

Another probable limitation can be that some new users can sign up account around same time from same region. So they will form group among themselves (may be legally) and can continue activity among only themselves. With a good choice of threshold, after some reasonable time they will be caught as voting group. But they are from a legal group.

Bibliography

[1] Cristopher Moore Aaron Clauset, Mark Newman, *Finding community structure in very large networks*, Physical Review E **1** (2004), no. 7.

[2] Simon Kang-Pu Liou Alex Pothen, Horst D, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM Journal on Matrix Analysis and Applications **11** (1990), no. 3.

[3] Alex M Andrew, *Information theory, inference, and learning algorithms, cambridge university press, cambridge*, **22** (2004).

[4] S Lin B W Kernighan, *An efficient heuristic procedure for partitioning graphs*, Bell Sys. Tech. J. **291** (1970), no. 18.

[5] Jonsson Barse, Kvarnstrom, *Synthesizing test data for fraud detection systems*, Annual Computer Security Applications Conference (2003), no. 384, 12.

[6] Hand Bolton, *Unsupervised profiling methods for fraud detection*, Credit Scoring and Credit Control VII (2001).

[7] Cécile Bothorel, Juan David Cruz, Matteo Magnani, and Barbora Micenková, *Clustering attributed graphs: models, measures and methods*, CoRR **abs/1501.01676** (2015).

[8] Golden Levine Alpert Brockett, Derrig, *Fraud classification using principal component analysis of ridits*, Journal of Risk and Insurance **69** (2002), no. 341, 31.

[9] Shawe-Taylor Burge, *An unsupervised neural network approach to profiling the behaviour of mobile phone users for use in fraud detection*, Journal of Parallel and Distributed Computing **61** (2001), no. 915, 11.

[10] Prodromidis Stolfo Chan, Fan, *Distributed data mining in credit card fraud detection*, IEEE Intelligent Systems **14** (1999), no. 67, 8.

[11] Volinsky Cortes, Pregibon, *Computational methods for dynamic graphs*, Journal of Computational and Graphical Statistics **12** (2003), no. 950.

[12] Wills Cox, Eick, *Visual data mining: Recognising telephone calling fraud*, Data Mining and Knowledge Discovery **1** (1997), no. 225, 7.

[13] Sanchez Cruz Dorronsoro, Ginel, *Neural fraud detection in credit card operations*, IEEE Transactions on Neural Networks **8** (1997), no. 827, 8.

[14] Norton Ezawa, *Constructing bayesian networks to predict uncollectible telecommunications accounts*, IEEE Expert (1996), no. 45, 7.

[15] M Sami F Luccio, *On the decomposition of networks in minimally interconnected subnetworks*, IEEE Transactions On Circuit Theory **16** (1969), no. 164.

[16] Stine Foster, *Variable selection in data mining: Building a predictive model for bankruptcy*, Journal of American Statistical Association **99** (2004), no. 303, 11.

[17] Illes Farkas-Tamas Vicsek Gergely Palla, Imre Derenyi, *Uncovering the overlapping community structure of complex networks in nature and society.*, Nature **8** (2005), no. 814.

[18] Reilly Ghosh, *Credit card fraud dection with a neural network*, Hawaii International Conference on Systems Science **3** (1994), no. 621, 10.

[19] Yao He, Graco, *Application of genetic algorithms and k-nearest neighbour method in medical fraud detection*, SEAL (1999), no. 74, 8.

[20] Tresp Hollmen, *Call-based fraud detection in mobile communication networks using a hierarchical regimeswitching model*, Advances in Neural Information Processing Systems (1998).

[21] D. Hughes, *Random walks and random environments*, Bulletin of Mathematical Biology **1** (1996), no. 598, 2.

[22] R D Luce, *Connectivity and generalized cliques in sociometric group structure*, Psychometrika **15** (1950), no. 169, 2.

[23] J. B. MacQueen, *Some methods for classification and analysis of multivariate observations. in cam lml. and neyman l.*, Berkeley Symposium on Mathematical Statistics and Probability (1967), no. 281, 17.

[24] Vanschoenwinkel Manderick Maes, Tuyls, *Credit card fraud detection using bayesian and neural networks*, International NAISO Congress on Neuro Fuzzy Technologies.

[25] M. Girvan Mark Newman, *Finding and evaluating community structure in networks*, Physical Review E **69** (2004), no. 2.

[26] Netmap, *Fraud and crime example brochure*, 2004.

[27] Girvan Newman, M.E.J., *Finding and evaluating community structure in networks*, Physical review e **2** (2004), no. 69.

[28] Mark Newman, *Fast algorithm for detecting community structure in networks*, Physical Review E **69** (2004), no. 6.

[29] Klaus Nordhausen, *The elements of statistical learning: Data mining, inference, and prediction, second edition by trevor hastie, robert tibshirani, jerome friedman*, International Statistical Review **77** (2009), no. 3.

[30] Lee Phua, Alahakoon, *Minority report in fraud detection: Classification of skewed data*, SIGKDD Explorations **6** (2004), no. 50, 10.

[31] Pascal Pons, *Detection de communautes dans les grands graphes de terrain.*, PhD thesis (2007).

[32] A D Pherry R D Luce, *A method of matrix analysis of group structure*, Psychometrika **15** (1949), no. 169, 2.

[33] S SCHAEFFER, *Graph clustering*, Computer Science Review **1** (2007), no. 27, 38.

[34] Sherman, *Fighting web fraud*, Newsweek (2002).

[35] Brian L Foster Stephen B Seidman, *A graph-theoretic generalization of the clique concept*, Mathematical Sociology **6** (1978), no. 139, 15.

[36] Paul R Shirley Stephen P Borgatti, Martin G Everett, *lambda sets and other cohesive subsets*, Social Networks **12** (1990), no. 337.

[37] Pan Syeda, Zhang, *Parallel granular neural networks for fast credit card fraud detection*, IEEE International Conference on Fuzzy Systems (2002).

[38] D. Wagner U. Brandes, M. Gaertler, *Experiments on graph clustering algorithms*, Europ. Symp. Algorithms (2003), no. 568, 2.

[39] Faust Katherine Wasserman, Stanley, *Social network analysis: Methods and applications. structural analysis in the social sciences*, Cambridge University Press **8** (1994), no. 8, 1.

[40] F. Y. Wu, *The potts model*, Reviews of Modern Physics **54** (1982), no. 1.

[41] Williams Milne Yamanishi, Takeuchi, *On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms*, Data Mining and Knowledge Discovery **8** (2004), no. 275, 26.