# Mining Direct Antagonistic Communities in Explicit Trust Networks

David Lo, Didi Surian, Kuan Zhang[*], and Ee-Peng Lim
School of Information Systems, Singapore Management University
Singapore
{davidlo,didisurian,eplim}@smu.edu.sg, zkspirit2@gmail.com

## ABSTRACT

There has been a recent increase of interest in analyzing trust and friendship networks to gain insights about relationship dynamics among users. Many sites such as Epinions, Facebook, and other social networking sites allow users to declare trusts or friendships between different members of the community. In this work, we are interested in extracting direct antagonistic communities (DACs) within a rich trust network involving trusts and distrusts. Each DAC is formed by two sub-communities with trust relationships among members of each sub-community but distrust relationships across the sub-communities. We develop an efficient algorithm that could analyze large trust networks leveraging the unique property of direct antagonistic community. We have experimented with synthetic and real datasets (myGamma and Epinions) to demonstrate the scalability of our proposed solution.

## Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Sociology*

## General Terms

Algorithms, Experimentation

## Keywords

Direct antagonistic community, Mining maximal bi-cliques, Signed social network

## 1. INTRODUCTION

Each of us forms trust and distrust relationships with others. With these relationships, communities are formed. At times, due to the nature of human interactions, some communities exhibit antagonistic behaviors among their members. Examples of such communities are many including social groups that hold differing opinions on topics such as industrialism vs. conservation, and formal organizations that are direct competitors in a market.

Several researchers have studied the nature of antagonistic communities [6, 5, 7]. It is well known that individual level antagonism exists in any social network. However, the existence of well "organized" form of antagonism between sub-groups should be detected in an early stage because they are potentially detrimental to the productivity and harmony of the community. Moreover, identification of antagonistic communities in the social networks could potentially open a path to further study about the structure of the community, the evolution of the community, etc.

Antagonistic communities on the other hand are not always bad. In some cases, they are actually welcomed. In massively multiple player online games, gamers are expected to form groups to fight with gamers that do not belong to the same groups. Here, a game's success depends very much on antagonistic behaviors among different gamer groups. The more antagonistic, the more challenging is the game.

In this study, our goal is to discover antagonistic communities automatically from their explicit trust and distrust relationships. We design a new graph mining algorithm to mine antagonistic communities by leveraging past work on computation of strongest connected components and bi-cliques. We take as an input a large graph of trusts and distrusts. From this graph, we would like to extract all pairs of user groups where within each pair, each group forms a connected trust network within themselves but distrusts members of the other group.

The contributions of this work are as follows:

1 We propose a new problem of mining antagonistic communities based on their explicit trust and distrust links.
2 We build a novel algorithm by using existing building blocks that have been shown to scale to large dataset hence enabling our approach to scale too.
3 We experiment our approach to extract antagonistic communities from real social networks, including a trust-distrust network and a friend-foe network.

The structure of this paper is as follows. Section 2 describes related work. Section 3 describes some preliminary definitions. Section 4 describes our approach. Experiments are presented in Section 5. We conclude and discuss future work in Section 6.

[*]The work was done while the author was with School of Information Systems, Singapore Management University

## 2. RELATED WORK

Vuong *et al.* [13] investigated content deletion between any two Wikipedia users as a form of disputes between them, and developed models to determine the degrees of controversy of users and articles using the dispute information. Different from Vuong *et al.*'s work, we mine direct antagonistic communities from networks with trust-distrust/friend-foe links.

Social network researchers have conducted a number of studies in signed networks (i.e., networks with both positive and negative relationships) in order to give more understanding about interaction between people in a community. The social balance theory developed for signed networks identifies triads with all positive relationships and triads with only one positive and two negative relationships as the balanced structure constructs [9]. Triads of other forms are known to be unbalanced. Dorean [8] studied how to partition signed social network. Yang *et al.* [14] mined communities in signed network using a heuristic clustering approach. In this work, we capture a set of people that are linked together by trust relationships and they also consistently oppose a common set of "enemies" in online social networks.

There are also several researchers who have studied intergroup antagonism concept [6, 7]. This body of work however has not been widely validated on large online social networks.

The enumeration of bi-cliques from graph data has been studied before. Including in this body of work are the work by Makino and Uno in [11] and Li *et al.* in [10]. The work by Li *et al.* in [10], to the best of our knowledge, is the latest in the series. They proposed a mapping between maximal bi-clique mining to frequent itemset mining problem [4]. We make use of their translation and extend their technique to mine for Direct Antagonistic Communities (DACs). A DAC is not a bi-clique, but some constraints within a DAC, in particular distrust relationships could be mapped to the problem of finding bi-cliques within a dataset. We use specific nature of DACs to prune additional nodes. The resultant technique scales well to mine from large real networks at low minimum size threshold within 15 minutes.

The closest to our work is the very recent work by Zhang *et al.* in [15]. Zhang *et al.* proposed an approach to mine for antagonistic communities based on rating data. Different from Zhang *et al.*'s work, in this work we mine for antagonistic communities based on *explicit* trust relationships. We believe *explicit* trust relationships are more reliable than common or differing ratings. A user could trust another user although they might have different "taste" on some common items of interest. Also, due to the different nature of the problem, there is a need to develop a new algorithm to mine for *direct* antagonistic communities from *explicit* trust networks.

## 3. PRELIMINARIES & PROBLEM DEFN.

In this section, we first describe preliminary concepts and definitions on graphs and frequent pattern mining. We then formalize some new definitions and our problem statement.

### 3.1 Preliminaries

Some standard definitions of graph, strongly connected subgraphs, strongly connected components, and bi-cliques are given in Definitions 1, 2, 3, 4 respectively.

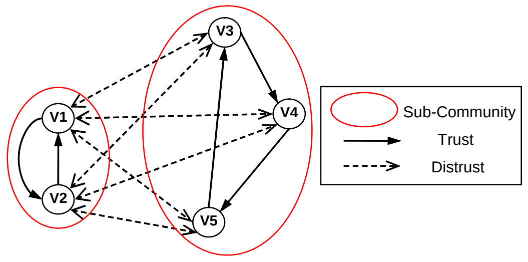


**Figure 1: A Direct Antagonistic Community.**

DEFINITION 1 (**Graph**). *A graph is composed of a set of nodes and edges and is denoted as G=(N,E). An edge is a mapping from one node to another node.*

DEFINITION 2 (**Strongly Connected Subgraphs**). *A strongly connected subgraph (SCS) is a sub-graph G' in a larger graph G where: For each node n' in G', there exists a series of edges in G' connecting n' to every other node in G'.*

DEFINITION 3 (**Strongly Connected Component**). *A strongly connected component (SCC) is a strongly connected sub-graph that is maximal in size.*

DEFINITION 4 (**Bi-Cliques**). *A bi-clique is a graph whose nodes could be decomposed of two sets of nodes where:*

1. *There are no edges among the nodes in each set*
2. *Each node is connected to every node in the other set.*

*We denote a bi-clique as (L,R), where L and R are the two sets of nodes having the characteristics described above.*

Next, we describe some preliminary definitions of transaction database, mapping function, itemsets, frequent itemsets and closed patterns in Definitions 5, 6, 7, 8 respectively. These terms are commonly used in frequent itemset mining first proposed in [4].

DEFINITION 5 (**Transaction DB & Mapping Func.**). *A transaction is a set of items from a domain D. A transaction database DB consists of a bag of transactions. Let map(S) be a mapping between a set of items S to the identifiers of the transactions in the DB containing S.*

DEFINITION 6 (**Itemset Patterns**). *An itemset pattern is a set of items. Consider a transaction database DB, the support of a itemset pattern P, is the number of transactions in DB that are super-sets of P. The support of P is denoted as sup(P).*

DEFINITION 7 (**Frequent Itemsets**). *An itemset P is a frequent itemset with respect to a transaction database DB and a minimum support threshold min_sup if sup(P) > min_sup.*

DEFINITION 8 (**Closed Patterns**). *An itemset P is a closed pattern, if P is frequent and there is no P' where P' $\supsetneq$ P and sup(P') = sup(P).*

### 3.2 Definitions & Problem Statement

We take as input a network of users expressing trusts/friendships and distrusts/foe relationships among themselves. We refer to this network as a trust-distrust (T-D) network defined in Definition 9.

Definition 9 (**Trust-Distrust Network**). *A trust-distrust (T-D) network is a graph whose nodes represent individuals and edges represent trust relationships among them. The edges are directed and are labeled with either: trust (T) or distrust (D). The nodes are labeled with the identifiers of respective individuals. A trust-distrust network could then be denoted as $G=(N,E,N_L,E_L)$ where N, E, $N_L$, and $E_L$ correspond to the nodes, edges, a mapping from nodes to labels, and a mapping from edges to labels respectively.*

Our goal is to mine a set of communities of antagonistic groups expressing explicit trust among themselves and distrust with members of the opposing group/sub-community. We refer to these communities as direct antagonistic communities defined in Definition 10.

Definition 10 (**Direct Antagonistic Community**). *A Direct Antagonistic Community (DAC) is composed of two sub-communities L and R. L and R are both SCSs with respect to the directed trust edges. Furthermore L and R form a bi-clique considering bidirectional distrust edges.*

An example of such Direct Antagonistic Community (DAC) is shown in Figure 1.

We are interested in DAC obeying a minimum size requirement, i.e., $|L| \geq min\_size$ and $|R| \geq min\_size$. We refer to such DACs as *significant* DACs. In Figure 1, the DAC example is significant if the minimum size threshold is set at 2; it would not be significant if the minimum size threshold is set at 3.

We next introduce the concept of *redundant* DACs, but first we need to describe *sub-bi-clique* operation. This is defined in Definition 11.

Definition 11 (**Sub-Bi-Clique**). *Consider a bi-clique C = (L,R). We define a sub-bi-clique of C, as a bi-clique C' = (L',R') where either $L' \subseteq L$ and $R' \subset R$, or $L' \subset L$ and $R' \subseteq R$. A sub-bi-clique of a bi-clique is a bi-clique.*

All sub-bi-cliques of a significant DAC are potentially significant DACs. Thus to prevent an explosion on the number of DACs, we mine only a compact representation of DACs. Given a set of mined DACs, we define redundant ones based on Definition 12. Only non-redundant DACs would be mined.

Definition 12 (**Redundant DAC**). *Consider a set of DACs ASET. One DAC a in ASET is deemed as* redundant *iff there exists another DAC a', where considering the bidirectional distrust edges, a is a sub-bi-clique of a'.*

With the above concepts and definitions, our problem definition is as follows:

**Problem Definition.** Given a trust-distrust network and a minimum size threshold *min_size*, find all significant non-redundant DACs.

## 4. MINING ANTAGONISTIC COMMUNITIES

In this section, we describe some properties of direct antagonistic communities and present our algorithm to mine them.

### 4.1 Properties

We use three properties in our mining algorithm outlined below. First, Property 1 describes a rule governing a node's membership to a significant DAC.

Property 1 (**Membership**). *Consider a node n in graph G, if n is not a part of any SCSs of size min_size, n could not be a part of any significant DACs.*

Proof. *From Definition 10, each sub-community in the DAC must be a SCS of size at least min_size. Hence, such a node n could not be a part of any DACs.* □

Next, Property 2 describes the relationship between a strongly connected subgraph and a strongly connected component in a trust-distrust network.

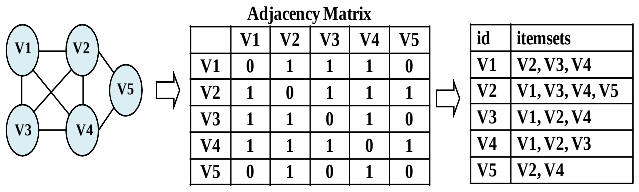
Property 2 (**SCS and SCC**). *Every Strongly Connected Subgraph (SCS) must be a part of a Strongly Connected Component (SCC).*

Proof. *From Definitions 2 & 3, a SCS could either be a SCC, or there is a super-graph of SCS which is a SCC.* □

We now define a new operation to convert a graph to a transaction database.

Definition 13 (**Graph to Transaction DB**). *The GTD operation converts a graph G to a transaction database DB by creating a new set of transactions $t = \{g' | (g,g') \in G.Edges\}$ for each node g in G and affixing the identifiers of g and g' to t. The resultant set of transactions is the result of the operation GTD(G).*

Figure 2 illustrates the GTD operation to derive a transaction database. With this operation, the duality between bi-cliques and closed patterns is established by Property 3.



Adjacency Matrix

| | V1 | V2 | V3 | V4 | V5 |
|---|---|---|---|---|---|
| V1 | 0 | 1 | 1 | 1 | 0 |
| V2 | 1 | 0 | 1 | 1 | 1 |
| V3 | 1 | 1 | 0 | 1 | 0 |
| V4 | 1 | 1 | 1 | 0 | 1 |
| V5 | 0 | 1 | 0 | 1 | 0 |

| id | itemsets |
|---|---|
| V1 | V2, V3, V4 |
| V2 | V1, V3, V4, V5 |
| V3 | V1, V2, V4 |
| V4 | V1, V2, V3 |
| V5 | V2, V4 |

**Figure 2: Graph to Transaction DB Operation**

Property 3 (**Bi-cliques and Patterns: Duality**). *Consider a graph G and a transaction database GTD(G). The set of all bi-cliques correspond to the set $\{(c,map(c)) | c \in CLS\}$ where CLS is the set of all closed patterns in GTD(G).*

Proof. *The above property has been proven in [10].* □

### 4.2 Proposed Algorithm

A Direct Antagonistic Community (DAC) has two basic requirements based on the trust and distrust relationships. On one hand, each community must form trust network in the form of strongly connected component. On the other hand, members of one community must form distrust relationships with all members of the other community. To mine for DACs, we perform the following steps:

1. Project input trust-distrust network $g$, to a graph $g_t$ keeping only trust edges in graph $g$.

2. Extract SCCs from $g_t$ of size more than the minimum support size *min_size*. These are candidate communities of DACs. Nodes that are not part of SCCs of size at least *min_size* could not be part of any DACs (see Properties 1 & 2). We keep the set of nodes $N_+ = \{n|n$ is a node in the identified SCCs$\}$.

3. Project the input trust-distrust network $g$, to a graph $g_d$ keeping only nodes in $N_+$ and bidirectional distrust edges.

4. Identify the set of maximal bi-cliques BCQ from $g_d$ using Property 3.

5. For each bi-clique in BCQ with set of nodes $n_b$, project the input trust-distrust network $g$, to a graph $g_{n_b}$ keeping only nodes in $n_b$ and their trust edges. Find SCCs from the projected network $g_{n_b}$. Each SCC satisfying *min_size* is a part of a DAC.

6. Eliminate redundant DACs. There could still be redundant DACs at the end of step 5. This is the case even after we mine for maximal bi-cliques of distrust at step 4, the DACs are sub-bi-cliques of the maximal one. We iterate through the set of DACs generated at step 5 and remove redundant ones based on Definition 12.

As a running example, we consider the graph shown in Figure 3 (Left). The graph contains trust edges (denoted as solid arrows) and bi-directional distrust edges (denoted as dashed bi-directional arrows).

**Pruning by Trust: Steps 1 & 2.** First, we prune candidate nodes based on trust relationships. Distrust edges are removed from the projected graph. Based on this trust graph, our goal is to throw away nodes which are not part of any large enough trust networks. Due to the nature of the trust network, the number of links of the nodes in the network follows power law, i.e., most nodes are not connected to any other nodes. Hence, a large number of nodes could be removed from the consideration.

To realize this goal, we employ Tarjan's algorithm [12], that could compute maximal SCCs by a single depth-first search pass on the trust network. Hence, it is very scalable as the runtime cost is linear to the size of the graph. We extract nodes that are part of a maximal SCC with size $\geq$ *min_size*.

Example.Consider *min_size* threshold being set to 2. From the example T-D network described in Figure 3 (Left), there are two SCCs. The first SCC consists of the set {V1,V2}. The second consists of the set {V4,V5,V6,V7}. The two SCCs are drawn in Figure 3 (Right). Hence we only retain nodes in the set {V1,V2,V4,V5,V6,V7} to be considered in the next step.

**Pruning by Distrust: Steps 3 & 4.** At these steps, we focus on the *strong* (i.e., bi-directional) distrust relationships. We project the input trust network, by removing trust edges and non bi-directional distrust edges. Two sub-communities in a direct antagonistic community must form a bi-clique with respect to the bi-directional distrust edges.

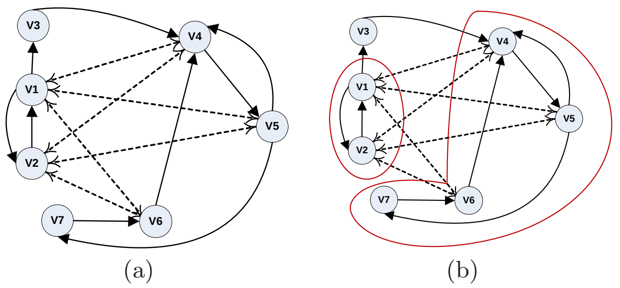


**Figure 3: Running Ex.: T-D Network (Left) & Two SCCs (Right)**

| TID | Itemset |
|---|---|
| V1 | {V4,V5,V6} |
| V2 | {V4,V5,V6} |
| V4 | {V1,V2} |
| V5 | {V1,V2} |
| V6 | {V1,V2} |
| V7 | {} |

**Table 1: Running Ex.: Transaction DB**

To realize the goal, we adapt a recent algorithm in [10] that extracts maximal bi-cliques from a graph following Property 3. The algorithm would return all maximal bi-cliques from the input bi-directional distrust network.

Example. Consider the distrust network projection on nodes {V4,V5,V6,V7} we come the network shown in Figure 4 (Left). This network is then converted to a transaction database shown in Table 1. Mining for closed patterns from this transaction database would result in the set of patterns: {{V1,V2}, {V4,V5,V6}}. This would correspond to one maximal bi-clique with L = {V1,V2} and R = {V4,V5,V6}.

**Formation of DACs : Step 5.** Each maximal bi-clique mined at step 4 is not necessarily a DAC as each of the two sets in the bi-clique is not necessarily a connected trust community. A bi-clique could map to 0 or more DACs.

Following Definition 11, every sub-biclique of a bi-clique is a bi-clique and hence satisfies the distrust requirement. Hence, we could extract sub-bi-cliques SBQ from each bi-clique in which each of the two sets of nodes forms a SCS of size larger than *min_size*.

To realize this, we process each bi-clique BCQ identified in step 4. For each of the two sets of nodes in BCQ, i.e., BCQ.L and BCQ.R, we find SCSs on the projected trust network containing nodes in BCQ.L/BCQ.R. These operations would result in two sets of SCSs. Pairing one SCS from one set with another from another set, would form a DAC which could then be included in the final result.

Example. Considering the maximal bi-clique with L = {V1, V2} and R = {V4,V5,V6}, the projection over the trust-distrust network is shown in Figure 4 (Right). From this projection, we could identify a DAC consisting of nodes {V1,V2} which is opposing nodes {V4,V5}. This DAC is shown by the red circles in the figure.

**Removal of Redundant DACs : Step 6.** Usually, there are no or few redundant DACs left at the end of step 5. The running example is one of such cases. However, there exist corner cases where redundant DACs are present. This is the case as mined DACs are sub-bi-cliques of the maximal bi-cliques mined at steps 3 & 4. We remove redundant DACs

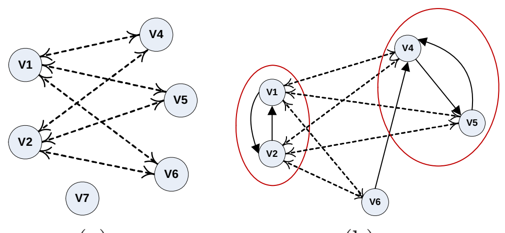


**Figure 4: Running Ex.: Distrust Projected Network (Left) & Resultant Projected Network and The Mined DAC (Circled in Red) (Right)**

by analyzing the list of DACs mined at step 5 and detect for redundancies based on Definition 12. We do this by comparing each DAC with every other larger DAC mined at step 5; each of these comparisons simply involves checking for subset relations among sub-communities of two DACs which is sufficient to decide whether one is a sub-bi-clique of the other.

We invite interested readers to [1] to investigate a corner case for T-D network where redundant DACs still exist after step 5 of the algorithm.

**Pseudocode.** Our algorithm's pseudocode is shown in Figure 5. At line 1, we perform graph projection and extract only the trust network from the input T-D network. At lines 2-3, we perform SCC computation and extract nodes that participate in an SCC of size at least *min_size*. At lines 4, we extract bi-directional distrust network from the input T-D network. This network is then converted to a special transaction database, and bi-cliques are identified via a closed itemset mining algorithm (lines 5-6). At lines 7-16, we check each bi-clique if it corresponds to zero, one, or more direct antagonistic communities of size at least *min_size*. At lines 10-11, we construct SCCs from the left and right set of nodes of each bi-cliques considering only trust links. We remove SCCs with size less than *min_size* at line 12. These SCCs are then composed to form direct antagonistic communities (lines 13-15). A non-redundancy check is performed at line 16.

## 4.3 Baseline Algorithm

The search space of all possible DACs is the set of all possible combinations of members in the T-D Network. As a baseline algorithm we perform a standard depth-first search traversal of this search space. We start with sets of users with one member and then grow each of these sets by adding additional members one by one. At each step of the process, we check if the set is a DAC. Due to space limitation, we present our baseline algorithm in [1].

## 5. EXPERIMENTS & ANALYSIS

To evaluate the scalability and efficacy of our approach, we experiment with both synthetic and real datasets. We build our own synthetic trust network generator that generates networks of different sizes for scalability test. We also consider two real datasets, a trust/distrust network from Epinions [3] and a friend/foe network from myGamma mobile social network site [2]. All these networks include both positive and negative edges.

The Epinions and myGamma datasets have 56,371 and 629,086 user nodes respectively, and about 250K and 8.1M

```
Procedure Mine Direct Antagonistic Communities
Inputs:   $G$: Trust-Distrust Network;
          $min\_size$: Minimum size threshold;
Outputs: Direct antagonistic communities with each opposing
         sub-community's size $\geq min\_size$;
Method:
1:  Let $G_t$ = Project trust network from $G$;
2:  Let SCCList = Get maximal SCCs from the graph $G_t$
                  by running [12];
3:  Let $N_+ = \{n \mid n \in s1 \wedge s1 \in SCCList \wedge |s1| \geq min\_size\}$;
4:  Let $G_d$ = Project bi-directional distrust network in $G$
              for nodes in $N_+$;
5:  Let $T_{Dis}$ = GTD($G_{Dis}$);
6:  Let CP = Mine for closed itemsets from $T_{Dis}$ with
              minimum support = $min\_size$;
7:  For each p in CP
8:    If $|p| \geq min\_size$
9:      Let BC = Form bi-clique (p,map(p));
10:     Let LT = Construct trust SCCs from nodes in BC.L;
11:     Let RT = Construct trust SCCs from nodes in BC.R;
12:     Remove SCCs from LT and RT with size $< min\_size$;
13:     For each pair $l \in$ LT and $r \in$ RT
14:       Create a new DAC adc from l and r;
15:       Add adc to Result;
16: Remove redundant DACs from Result;
17: Output Result;
```

**Figure 5: Mine Direct Antagonistic Communities**

| Network | # Nodes | # +ve Links | # −ve Links |
|---|---|---|---|
| Epinions | 47,910 | 208,531 ($\sim$ 99.49%) | 1,071 ($\sim$ 0.51%) |
| myGamma | 611,211 | 7,563,927 ($\sim$ 92.99%) | 570,066 ($\sim$ 7.01%) |

**Table 2: Real Datasets: Statistics**

links. As not all users have links, we remove those without trust/friendship or distrust/foe links and obtain the numbers of nodes as shown in Table 2. Both real datasets have more than 90% positive links. myGamma is has roughly 12.7 times more nodes and 38.8 times more links compared with Epinions.

We run our algorithm on an Intel(R) Core(TM) 2 2.13GHz PC with 2 GB of RAM running 32-bit Windows XP, Service Pack 3. The algorithm is written in Visual C#.Net.

We describe our scalability experiments in the following subsections. We have also performed an efficacy study; however, due to space limitation, we move its description to a technical report [1].

## 5.1 Scalability Experiment using Synthetic Datasets

We develop a synthetic trust network generator that uses three input parameters, i.e., $|N|$ (i.e., the number of nodes in the graph), $|E|$ (i.e., the number of edges in the graph), and $\tau$ (proportion of distrust links). By varying the three parameters, networks of different sizes (i.e., $|N|$ and $|E|$) and composition of trust and distrust links ($\tau$) can be generated. We present detail steps of our synthetic network generation process in [1].

Five synthetic trust networks have been created with: (a) $|N|$=100, $|E|$=6,000, $\tau$=0.4; (b) $|N|$=150, $|E|$=13,500, $\tau$=0.4; (c) $|N|$=500,000, $|E|$=15,000,000, $\tau$=0.1; (d) $|N|$=800, 000, $|E|$=38, 400,000, $\tau$=0.1; (e) $|N|$=1,000,000, $|E|$=6,000, 000, $\tau$=0.1. We mine DACs from these synthetic T-D networks with *min_size* set to 2 using our proposed algorithm

| Dataset | Proposed Algorithm | Baseline Algorithm |
|---|---|---|
| (a) | 0.297 s | 372.930 s |
| (b) | 0.937 s | 2,985.169 s |
| (c) | 127.333 s | >24 hours |
| (d) | 360.771 s | >24 hours |
| (e) | 812.989 s | >24 hours |

**Table 3: Running Time for Synthetic Network**

| | Epinions | | myGamma | |
|---|---|---|---|---|
| min_size | Time (s) | $\lvert DACs\rvert$ | Time (s) | $\lvert DACs\rvert$ |
| 1 | 3.875 | 562 | 844.727 | 50455 |
| 2 | 1.703 | 21 | 82.958 | 50 |
| 3 | 1.641 | 7 | 78.397 | 4 |
| 4 | 1.641 | 1 | 76.95 | 0 |
| 5 | 1.609 | 0 | 81.164 | 0 |

**Table 4: Time & $|DACs|$: Epinions and myGamma**

and our baseline algorithm (with maximum DAC size threshold set at 5[1]). We show the running time for each synthetic trust network using our proposed algorithm and our baseline algorithm in Table 3.

From Table 3, we show that our approach is able to efficiently process a large network of 1 million nodes using a low *min_size* threshold within 14 minutes. We also observe that the running time increases with the network size. This is particularly due to the computational overhead of mining bicliques in the algorithm. We also notice that our proposed approach is more than 1,000 times faster than the baseline approach. The baseline approach is not able to complete within one day for the larger T-D networks (i.e., network (c)-(e)). This result shows the power of our proposed pruning strategies.

## 5.2 Scalability Experiments using Real Datasets

The running times for mining DACs on Epinions and myGamma datasets at various minimum size thresholds using our proposed approach are shown in Table 4. Our baseline approach is not able to complete for these datasets. We show the number of direct antagonistic communities mined along with the time needed to mine them.

As shown in Table 4, our algorithm completes extracting all DACs from the Epinions dataset within 3.875 seconds. We do not find any large DACs with sub-community size $\geq 5$, as there are zero antagonistic communities when the size threshold is set to be 5. On the myGamma dataset, the running time ranges from 81.164 seconds (for *min_size*=5) to 844.727 seconds (for *min_size*= 1). The running time required is larger than that for Epinions data due to larger data size. We also notice that there are more DACs mined from myGamma. However, interestingly, proportion-wise, Epinions dataset has DACs of slightly larger sizes. myGamma does not have any DACs of size 4 and above.

## 6. CONCLUSION AND FUTURE WORK

In this study, we analyze large graphs of trust and distrust to extract direct antagonistic communities (DACs). Within a DAC there are two opposing sub-communities. We propose a new framework to extract DACs efficiently from large networks. Our approach consists of several steps including: pruning of nodes not involved in any trust network, detection of bi-directional distrust bi-cliques among two sets of users, formation of DACs from distrust bi-cliques, and detection of redundant DACs. Experiments have been conducted on both synthetic and real datasets. On the synthetic dataset our approach is able to scale to 1,000,000 nodes and 6,000,000 edges. Our approach could also run on two medium-large real datasets from Epinions and myGamma within a reasonable time. Our approach uses less than 5 seconds to extract DACs from Epinions dataset that has more than 250,000 links among more than 560,000 members. Moreover, our approach needs less than 15 minutes to deal with myGamma dataset that has more than 8 millions links among more than 600,000 members.

As future work, we plan to investigate the evolution of DACs as a social network changes over time. Also, we plan to investigate factors leading to the creation of antagonistic communities.

**Acknowledgement.** We would like to thank Paolo Massa for sharing his Epinions dataset. We also would like to thank myGamma for sharing their dataset with us, and National Research Foundation (NRF) (NRF2008IDM-IDM004-036) for funding the work.

[1]The largest size of maximal DACs reported by our approach in dataset (a) is 5 which we use to set this threshold.

## 7. REFERENCES

[1] http://www.mysmu.edu/faculty/davidlo/papers/LoSurianZhangLimTR.pdf.

[2] "Buzzcity - myGamma - the world's leading wireless community," http://www.buzzcity.com/f/mygamma.

[3] "Reviews from Epinions." http://www.epinions.com/.

[4] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *VLDB*, 1994.

[5] I. Dasgupta, "'living' wage, class conflict and ethnic strife," *Journal of Economic Behavior & Organization*, 2009, in press.

[6] I. Dasgupta and R. Kanbur, "Community and class antagonism," *Journal of Public Economics*, vol. 91, no. 9, pp. 1816–1842, Sep 2007.

[7] J. Denrell, "Why most people disapprove of me: Experience sampling in impression formation," *Psychological Review*, vol. 112, no. 4, pp. 951–978, 2005.

[8] P. Doreian and A. Mrvar, "Partitioning Signed Social Networks," *Social Networks*, vol. 31, pp. 1–11, 2009.

[9] D. Easley and J. Kleinberg, "Positive and negative relationships," in *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, 2010, pp. 119–152.

[10] J. Li, G. Liu, H.Li, and L. Wong, "Maximal biclique subgraphs and closed pattern pairs of the adjacency matrix: A one-to-one correspondence and mining algorithms." *TKDE*, vol. 19, no. 12, pp. 1625–1637, Dec 2007.

[11] K. Makino and T. Uno, "New algorithms for enumerating all maximal cliques," in *Proceedings of the 9th Scandinavian Workshop on Algorithm Theory*, 2004.

[12] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.

[13] B.-Q. Vuong, E.-P. Lim, A. Sun, M.-T. Le, H. Lauw, and K. Chang, "On ranking controversies in wikipedia: Models and evaluation." in *WSDM*, 2008.

[14] B. Yang and D.-Y. Liu, "A Heuristic Clustering Algorithm for Mining Communities in Signed Networks," *Journal of Computer Science and Technology*, vol. 22, pp. 320–328, 2007.

[15] K. Zhang, D. Lo, and E.-P. Lim, "Mining antagonistic communities from social networks," in *PAKDD*, 2010.