Discovering Influential Nodes from Trust Network

Sabbir Ahmed
University of Windsor
School of Computer Science
Windsor, Ontario

ahmedp@uwindsor.ca

C.I. Ezeife†
University of Windsor
School of Computer Science
Windsor, Ontario

cezeife@uwindsor.ca

ABSTRACT
The goal of viral marketing is that, by the virtue of mouth to mouth word spread, a small set of influential customers can influence more customers. Influence maximization (IM) task is used to discover such influential nodes (or customers) from a social network. Existing algorithms for IM adopt Greedy and Lazy forward optimization approaches which assume only positive influence among users and availability of influence probability, the probability that a user is influenced by another.

In this work, we propose the T-GT model, which considers both positive (trust) and negative (distrust) influences in social trust networks. We first compute positive and negative influences by mining frequent patterns of actions performed by users. Then, a local search based algorithm called mineSeedLS for node add, exchange and delete operations, is proposed to discover influential nodes from trust networks. Experimental results shows that our approach outperforms Greedy based approach by about 35%.

Categories and Subject Descriptors
H.2.8 [**Database Applications**]: Data Base Application – *Data Mining*.

General Terms
Algorithms, Measurement, Economics, Experimentation.

Keywords
Social Network, Trust Network, Viral Marketing, Data Mining, Submodular Function.

1. INTRODUCTION
Viral Marketing is the process of targeting the most influential users in the social network so that these customers can start a chain reaction of *influence* driven by word-of-mouth, so that with a small marketing budget a large population of a social network can be reached or *influenced*. For example a phone manufacturer wants to promote their new phone model and have limited budget for the marketing campaign.

To get maximum possible benefit out of the limited budget the company may want to choose a small group with largest influence, so that this small 'influential' group can influence greater number of potential customers. Selecting such influential nodes from a social network graph is an interesting research challenge that has received a good deal of attention in recent years [1]. Kempe et al. [6] formalize *Influence Maximization* problem in terms of *diffusion models* (definition below), such as Linear threshold (LT) model and Independent Cascade (IC) model adopted from mathematical sociology. They also presented a generalized model called General Threshold (GT) model which is an extension of IC and LT models.

Diffusion Model - A diffusion model, also known as *propagation model*, describes the entire diffusion process and determines which nodes will be activated due to the influence spread through the social network.

Diffusion models, in general, model the spread of influence or the diffusion process, through a social network represented by a directed graph $G(V,E)$, where V is the set of all users (also called nodes) in the social network and E is the set of directed edges between these nodes. In these diffusion models a node or user is said to be *active* if the node adopts a product (or performs an action) or *inactive* if the node does not adopt a product (or performs an action). Given a social network graph $G(V,E)$, a diffusion model M and an initial set of *active* vertices $S \subseteq V$, the *influence spread* of set S, denoted $\sigma_M(S)$, is the expected number of vertices to become active, under the influence of vertices in set S, once the diffusion process is over. All existing diffusion models, such as IC and LT [6], requires additional parameters for each directed edge in E, such as *influence probability,* along with social network graph $G(V,E)$, to determine or compute influence spread.

Influence probability – Given a social network graph $G(V,E)$, influence probability, denoted as $p(u,v)$ (such that $u \in V$, $v \in V$ and $(u,v) \in E$) is the probability of node v to perform some action under the influence of user u.

Using these notations Kempe et al. [6] defines the k-best influence maximization problem as follows:

Problem 1 - *Given a social network graph $G(V,E)$ along with influence probabilities of all edge in E, a diffusion model M and a number k such that $k \leq |V|$, find a set S such that $S \subseteq V$, $|S| \leq k$ and the influence spread, that is $\sigma_M(S)$, is maximum.*

SAC'13, March 18-22, 2013, Coimbra, Portugal.

† This research was supported by the Natural Science and Engineering Research Council (NSERC) of Canada under an operating grant (OGP-0194134) and a University of Windsor grant.

From viral marketing perspective the parameter k is the budget of how many individuals we want to target and is given as input by the end user. The selected set S is also referred to as "seed set". Kempe et al. [6] proves that the optimization problem (problem 1) is NP-hard. However, they show that $\sigma_M(.)$ function is submodular and monotone. This actually means that under IC and LT models, the effect or the *marginal gain* (expressed as $\sigma_M(S \cup \{v\}) - \sigma_M(S)$); of adding a new node to a seed set S is smaller than that of adding the same node to a subset of S. A submodular function σ is said to be monotone if we have $\sigma(S) \leq \sigma(S \cup \{v\})$ for all elements v and sets $S \subseteq V$. That is adding an element to a set does not decrease the value of the function $\sigma(.)$.

According to Nemhauser et al. [13] any submodular monotone function can be solved using natural greedy algorithm with a (1-1/e) approximation guarantee. That is due to the submodular and monotone property of $\sigma_M(.)$ function, the greedy solution will produce result which is at least 63% of the optimal. The greedy algorithm for influence maximization requires 2 inputs as follows [5][6]:

a) A directed social network graph *G(V,E)* and

b) Influence Probability of each edge in E.

A major issue that is largely ignored in the works in influence maximization, such as [2][6], is the question – *How we can compute Influence Probability?* In most of the works such as [2][6], it is assumed that the network itself and influence probabilities are known and given to the IM algorithm as input. A social network graph G can be easily constructed if the data is explicitly available [5]. However influence probabilities are not explicitly available. To tackle this issue researchers are now looking into ways to *mine* influence probabilities from Action Log of users in a social network [5]. Action log is a relation, Actions(User, Action, Time). It contains tuples [5], such as (*u, a, t*), which means that user u (such that $u \in V$) performed action a, at time t.

All the work in the area of influence maximization, such as [2][3][6], consider only *positive* influence among users in a social network. That is, techniques for influence maximization only consider how much a node has influence on another node to perform a certain task. However, in real life scenario a node can also have some degree of *negative* influence on another node, especially by a user who he/she does not trust. Furthermore, viral marketing is different from other strategies of marketing, because it is based on *trust* among close families and friends [3]. Existing diffusion models for IM are modeled such a way that a node's probability of performing an action (or adopting a product) will increase as the number of his/her friends performing the same action increases. However, we argue that, a node's probability of performing an action (e.g., Buy iPhone 4S) should also *decrease* if its distrusted neighbours, also buy iPhone 4S.

1.1 Contributions and Outline

Motivated by the issues discussed above, the problem we tackle is as follows:

Problem Definition – *Find Influential Nodes from a directed trust network graph, G(V,E) where every edge (u,v) of E is directed and labelled either positive (trust) or negative (distrust), and given an Action Log, Actions(User, Action, Time) such that every user u in User column of action log table is member of V.*

To solve the above problem we make the following contributions in this research:

1) We propose a new diffusion model named Trust-General Threshold (TGT) model which incorporates both positive and negative influence probabilities based on trust relationship among users in trust network.

2) Based on this new TGT model we propose a new influence maximization framework for trust network, called Trust-Influential Node Miner (T-IM), which takes trust network data and action log to find influential nodes.

3) To compute influence probabilities we propose to mine action log to find frequent patterns of action performed by trusted and distrusted neighbours and use it to compute both positive and negative influence probability using Bernoulli distribution.

4) We claim that influence spread under TGT model is *non-monotone* sub modular function and define the problem of finding influential nodes from trust network as *maximization of non-monotone submodular function problem*.

5) We propose a new algorithm, mineSeedLS, involving local search based on Lee et al. [9] to solve IM under our proposed TGT model.

6) Perform experiments and analysis of our proposed solution using real life data set collected from Epinions (www.epinions.com) [12] and Wikipedia (www.wikipedia.com). In terms of quality of influential nodes selected, our experiments shows that mineSeedLS outperforms greedy based solutions by almost 35%.

Note that the main problem tackled in this paper is to define and solve influence maximization considering both positive and negative influence among the users in a trust network to find influential nodes. Section 2 provides relevant background and related works. In section 3 we propose a new diffusion model, called Trust-General Threshold (TGT) Model, which is an extension of general threshold model by Kempe et al. [6]. In section 4 we provide the solution framework of the problem. Section 5 reports the results of our experiments and section 5 provides concluding remarks and future works in this area.

2. RELATED WORK

Domingos and Richardson in [4] and [11], introduce the problem of identifying influential customers by taking into account their network value. The network value of a customer in a social network is the profit due to additional sales to customers he or she may influence to buy [4][11]. Their goal is to provide a solid foundation for viral marketing problem, which exploits network value of customers.

Kempe et al. [6], motivated by the work of Domingos and Richardson [4][11], tackled the viral marketing problem using several commonly used diffusion models, such as the Linear Threshold, Independent Cascade and General Threshold Models. Using these models they formulated the viral marketing problem as the *influence maximization* problem which is a discrete optimization problem and proved it to be NP-Hard. However, they presented further proof to show that the influence maximization problem can be solved with good approximation using a natural greedy approach. This is due to the fact the influence spread function is submodular and monotone under proposed diffusion models [6].

In [6], Kempe et al. presented the greedy algorithm. The algorithm requires computing marginal gain of influence spread of every node *v*, $\sigma_M(S \cup \{v\}) - \sigma_M(S)$, in each iteration. The node with highest marginal gain is 'greedily' added to the seed set until number of nodes in seed set reaches *k*. Although simple, the greedy algorithm of Kempe et al. [6] is computationally expensive and not scalable to large social network. According to Chen et al. [2][3] the greedy algorithm would fail or become unfeasible to extract influential nodes from large social network graph with more than 500K edges even on modern server machine. The computationally expensive step of the greedy algorithm is where we select the node that provides the largest influence spread. To be able to do this we need to compute influence spread of each and every nodes, which can be quite time consuming if the number of nodes and edges in the social network graph is very large. Most of the work, such as work of Leskovec et al. [10] and of Chen et al. [2][3], that followed in the area of influence maximization attempt to tackle the efficiency issue of the greedy approach. However, this is to be noted that tackling scalability is not within scope of this research and so we keep the discussion on this issue limited.

Existing algorithms for influence maximization, such as Greedy [6] and 'Lazy Forward' [10] based algorithms, requires that the influence probability is known and given to the algorithm as input along with a social network graph. A social network graph can be easily constructed if the relationship (among users in a social network) data is explicitly available. However influence probabilities are not explicitly available. In most of the literature reviewed influence probabilities are assumed and given as input [5].

To tackle this, researchers are recently looking into data mining techniques [5] to mine influence from user's action log. These techniques in general takes Action Log, which is a relation Actions (User, Action, Time), along with a social graph *G*(*V, E*). Action log are extracted from log of user activity in a social network site databases. Tuples in Action log table contains, for e.g. (*u, a, t*), indicating that user *u* (such that $u \in V$) performed action *a*, at time *t*. Recent researches show that such action log can provide traces of influence among users in a social network [5]. For example if a user *v* rate "Mission Impossible" movie and later *v*'s friend *u* does the same, and then the action of rating the movie "Mission Impossible" propagates from user *v* to user *u*.

3. TGT MODEL

3.1 General Threshold Model

Recall that general threshold (GT) model [6] is a generalized model of LT and IC models which is defined as follows. Let us consider an inactive user *u* and the set of its active neighbors *S* (that is all nodes in *S* already performed certain task). To determine if *u* will activate (or perform the task), we first compute $p_u(S)$, which is the joint influence probability of *S* on *u* [5]. If $p_u(S) \geq \theta_u$, where θ_u is the activation threshold of user *u*, according to GT model we conclude that *u* activates [5]. The joint probability $p_u(S)$ is computed as follows [5][6], where $p_{v,u}$ is the influence probability of any node *v* on another node *u*:

$$p_u(S) = 1 - \prod_{v \in S}(1 - p_{v,u}) \qquad (1)$$

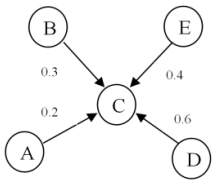

Figure 1: Example of social network graph with influence probabilities as weight on edges

3.2 Trust-General Threshold Model

In case of trust based network the above joint probability equation is appropriate only if we consider only trusted neighbours of node *u*. Let us consider two scenarios for node C in figure 1. Let us assume that node C trusts node E. Also node C distrusts node A. In the first scenario, let us consider node E gets activated at time *t*. Also let us assume the influence probability $p_{E,C}$=0.3. So according to equation 1, the probability of node C getting activated, i.e. $p_C(\{E\}) = 1 - (1 - 0.3) = 0.3$. In the second scenario, let us consider node E and A get activated at time *t*. Since node E is the only trusted neighbour of node C the probability of node C getting activated in second scenario is also, i.e. $p_C(\{E\}) = 1 - (1 - 0.3) = 0.3$. In the second scenario, we should also consider any negative influence on node C by node A, because C does not trust A. If few of trusted friends of a user adopt a product, the probability of him/her to also adopt a product should not stay the same if few distrusted user also adopt the product.

To accommodate such *negative* influence while computing $p_u(S)$, we introduce the notion of *negative influence probability*. Negative influence probability is the probability, denoted as $p^-_{v,u}$, of a user *u* not getting activated due to negative influence of node *v*. Here, we assume that $p^-_{v,u} = 0$ if node *u* trusts node *v*. That is, there is no negative influence on a node by any of its trusted neighbours. Similarly, we also assume positive influence probability, from here on denoted as $p^+_{v,u}$, is 0 if node *u* does not trust user *v*. That is, there is no positive influence by any of its distrusted neighbours.

Let S^+ be all the trusted active neighbours of node *u* and S^- be the distrusted active neighbours of node *u*. Let us say $S = S^+ \cup S^-$ .To determine whether *u* will activate given that all nodes in *S* are active, we first compute $p_u(S^+)$ and $p_u(S^-)$. To be able to do this we simply use the equation 1 individually for S^+ and S^-. That is:

$$p_u(S^+) = 1 - \prod_{v \in S^+}(1 - p^+_{v,u}) \qquad (2)$$

$$p_u(S^-) = 1 - \prod_{v \in S^-}(1 - p^-_{v,u}) \qquad (3)$$

$p_u(S^+)$ is the *positive* joint influence probability, which is the probability of node *u* performing an action under influence of its trusted neighbours. And $p_u(S^-)$ is the *negative* joint influence probability, which is the probability of node *u* not performing an action under negative influence of its distrusted neighbours. According to the proposed TGT model we say a node becomes active if:

$$p_u(S^+) > \theta^1 \text{ AND}$$

$$p_u(S^-) < \theta^2$$

Where θ^1 and θ^2 are thresholds that are chosen uniformly at random for each node. Thresholds, θ^1 and θ^2, essentially represents the latent tendencies of nodes to adopt (or not to adopt) a product when their trusted and distrusted neighbours do. These are randomly chosen to model the unavailability of these values in real life [8]. To find influential nodes based on the TGT model we need to estimate influence spread denoted as $\sigma_{TGT}(S)$**,** the expected number of nodes influenced by a given set of nodes. Influence spread of given set *S* is counted as follows [6][7]:

i. First a list of currently active nodes, *H*, is created. That is *H* initially consists of all the nodes in *S*

ii. Then for each neighbours of every nodes in *S*, positive and negative joint influence probabilities are computed according to equation 1 and 2.

iii. For every node *u*, outside *S*, that becomes active according to TGT model, is added to *H*.

iv. For each neighbour of *u* that became active in previous step, the positive and negative joint influence probabilities are computed according to equations 2 and 3.

v. If more nodes become active then, repeat step iii. If no more nodes become active then the diffusion process stops here. The influence spread, $\sigma_{TGT}(S)$**,** is basically the total number of nodes in *H.*

3.3 Influence Maximization under TGT

Influence maximization under existing diffusion models such as LT and IC can be solved with Greedy [6] or Lazy Forward optimization [10] with 63% approximation guarantee. However these approaches rely on the fact that the influence spread function, $\sigma(.)$, is monotone and submodular. In contrast the influence spread function, denoted as $\sigma_{TGT}(S)$, under the new proposed TGT model is non-monotone. That is, adding a node (or user) may not result in influence spread to increase in TGT model. To show this let us consider the following scenario. Let *S* is the initially activated seed set and $\partial(S)$ are the nodes that were successfully activated by the seed set *S.* That is the influence spread, $\sigma(S)$, is actually the number of nodes in $\partial(S)$ or $|\partial(S)|$. Now let us consider a node *w* that has a negative influence (due to distrust) on two nodes *u* and *v* which is in $\partial(S)$. Now adding *w* to S will cause the probability of *u* and *v* getting activated to decrease according to equation 4 and will not get activated. This will cause the influence spread of $S+\{w\}$, i.e. $\sigma_{TGT}(S+\{w\})$, to decrease as $|\partial(S)-2| < |\partial(S)|$. Therefore, we claim that influence spread function is non-monotone. So, to solve influence maximization under TGT model, the approximation guarantee by Greedy approaches by [6] and [10] is not applicable. However, we show that the spread function under TGT model is still sub modular. This claim based on the following theorem of Kleinberg [8].

Theorem 1 [8]: For any instance of the General Threshold Model in which all the threshold functions are submodular, the resulting influence spread function, $\sigma(.)$, is submodular.

Note that the threshold functions in TGT model are basically equations 2 and 3 which happen to be submodular [5]. Therefore, we define the problem of finding influential nodes from trust network as a problem of *maximizing non-monotone submodular function* under budget constraints. So, we use local search based algorithm by Lee et al. [9] to extract influential nodes from trust network. Local search algorithms are commonly used to solve computationally hard optimization problems that can be formulated as finding a solution which maximizes a criterion, among a number of solutions. Local search algorithms typically start with a small solution based on certain criteria. Then, it applies local changes to the current solution, such as adding an element or removing an element, until a solution deemed optimal is found or certain criteria (example *budget*) is met.

4. SOLUTION FRAMEWORK

We now discuss the overall proposed solution framework, called Trust-Influential Node Miner (T-IM), for discovering influential node from trust network. Following are the inputs to T-IM framework:

i. Action Log Table (Table 1) – Contains tuples, for example (*a, u, t)*, which indicates that node *u* performed action *a*, at time *t.*

ii. Trust Data (Table 2) – Contains tuples, for example (*u, v, trust*). If *trust* is 1 it indicates that node *u* trusts node *v*. If trust is -1 it indicates that node *u* does not trust node *v*.

iii. Budget – Number of influential nodes to be extracted.

Table 1: Action Log Table example

Action	***User***	***Time***
a	u1	5
a	u2	6
a	u3	8
a	u4	11
b	u2	4
b	u3	5
b	u5	8
c	u2	11
c	u5	12
c	u1	18

The proposed solution consists of following three main steps listed below. Note that in the rest of this paper and in the algorithms presented, we denote adding an element *v* to any set *S* by $S = S + \{v\}$. Similarly, we denote removing an element *v* from any set *S* by $S = S-\{v\}$. Also, *V-S* is the set of elements which are in set *V* but not in set *S*.

Step 1 – First, we construct a social network graph *G*(*V,E*) using the Trust Data table. For each tuple (*u, v, trust*) it adds nodes *v* and *u* to set *V* of social network graph *G*(*V,E*). It adds an edge (*v,u*) to set *E* of the social network graph. This is because if *u* trusts (or distrusts) *v* then there is an influence (positive or negative) of node *v* on node *u.* Example of the social network graph constructed from sample trust table 2 is shown in figure 2. The influence probabilities labeled in figure 2 is computed in the next step.

Table 2: Trust Data example

U	*V*	*Trust*
u1	u2	1
u1	u3	-1
u2	u1	1
u2	u4	-1
u2	u5	-1
u4	u1	1
u4	u2	-1
u5	u2	1
u5	u3	1

Step 2 – In this second step the action log is mined to extract patterns of actions, which are required to compute the influence probabilities. Then, influence probabilities are computed using these patterns (discussed in section 4.1) for each edge (*u,v*) in *E* of social network graph we constructed in Step 1.

Step 3 – This is the main and final step of our proposed framework which takes social network graph *G*(*V,E*), the trust matrix TM, and Influence Matrix IM to extract influential nodes. To mine influential nodes (or seed set) using local search based technique, we use algorithm called mineSeedLS. Detail of this algorithm is given in section 4.2.

4.1 Learning Influence Probability

To estimate these probabilities we use action log and extract pattern of user behavior. To do this we extract two types of frequent patterns from the action log. The first pattern we extract, we call it Positive Frequent Action Pattern, is the number of actions performed by any node *u* after the same actions were performed by all trusted neighbors of *u.* This is similar to extracting frequent item set in frequent pattern mining. In frequent pattern mining we are interested to find items which appear frequently in data.

Let us say a node *u* performs $A_{v.u}$ number of actions after its trusted neighbor *v* and node *v* performs total of A_v tasks in total. We compute positive influence probability of node *v* on node *u* by using dividing $A_{v.u}$ by A_v. In frequent pattern mining this is also known as *confidence* which is interpreted as the probability of *u* performing an action after *v.* Note that unlike traditional frequent pattern mining we are also interested in computing number of times a node *does not* perform a task after its distrusted neighbour. Therefore, we extract second pattern, we call it *Negative Frequent Action Pattern*, which counts the number of actions not performed by any node *u* after the same actions were performed by a distrusted neighbor of *u.* Let us say a node *u* does not perform $A'_{v.u}$ number of actions after its distrusted neighbor *v* and node *v* performs total of A_v tasks in total. We compute negative influence probability of node *v* on node *u* by using dividing $A'_{v.u}$ by A_v. To illustrate this, let us consider that in a trust social network, a node *u* trusts another node *v* and also distrusts another node *w.* Now, let us assume that according to action log node *v* performs a total of 3 actions. And out of these 3 actions 2 actions were performed by *u* after node *v* (trusted neighbor of *u*) performs these same actions. So, the probability of node *u* performing a task after node *v* performs the same action is 2/3 = 0.66. This is the positive influence probability of node *v* on node *u.* That is, $p^{+}_{v,u}$ = 0.66. Based on this, for any nodes *u* and *v,* if $A_{v.u}$ is the number of actions performed by *u* after node *v* and A_v is the number of actions performed by node *v* positive influence probability of node *v* on node *u* can be expressed as:

$$p^{+}_{v,u} = \begin{cases} 0 \; if \; u \; distrust \; v \\ \dfrac{A_{v.u}}{A_v} \; otherwise \end{cases}$$

Now let us further consider that a node *w* (distrusted neighbour of *u*) performs 4 tasks in total and out of this, only 1 task was performed by *u* after *w.* That is, *u* did not perform 3 out of 4 tasks performed by *w.* So, the negative influence probability of node *w* according Bernoulli distribution (citations please) is 3/4 = 0.75. Based on this, for any nodes *u* and *v,* if $A'_{v.u}$ is the number of actions not performed by *u* after node *v* and A_v is the number of actions performed by node *v,* negative influence probability of node *v* on node *u* can be expressed as:

$$p^{-}_{v,u} = \begin{cases} 0 \; if \; u \; trusts \; v \\ \dfrac{A'_{v.u}}{A_v} \; otherwise \end{cases}$$

Note that to compute influence probabilities as discussed above, we need to learn required parameters, such as $A_{v.u}$, $A'_{v.u}$, and A_v from action log. Once both positive and negative influence probabilities are learned from action log, we are in the position to discover influential nodes from trust network under TGT model.

Table 3: A_v of each node *v* from Action Log in table 1

v	u1	u2	u3	u4	u5
A_v	2	3	2	1	2

Table 4: $A_{v.u}$computed from Action Log in table 1

v	***u***	$A_{v.u}$
u2	u1	1
u2	u3	2
u1	u2	1
u1	u4	1
u2	u5	1
u3	u5	2

Table 5: $A'_{v.u}$ computed from Action Log in table 1

v	***u***	$A'_{v.u}$
u3	u1	2
u5	u3	2
u4	u2	1
u5	u2	2
u2	u4	2

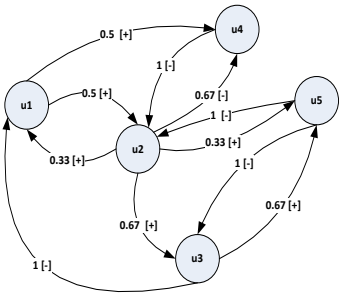

Figure 2: Social network graph (using trust data in table 2) where each edge is labeled with influence probabilities and indicates if it is positive '[+]' or negative '[-]'

4.2 Discovering Influential Nodes

Now we present the algorithm which mines for influential nodes, called mineSeedLS (Algorithm 1). It takes social network graph G(*V,E*) with influence probabilities (figure 2) and an integer *budget*. The algorithm returns set of influential nodes, *S,* such that *S* is a subset of *V* and |*S*|<= *budget.* The algorithm starts by initializing seed set *S* to NULL (Line 1). In line 2 the algorithm computes influence spread of each node *v* according to TGT model, $\sigma_{TGT}(\{v\})$. Note that the influence spread $\sigma_{TGT}(\{v\})$ is computed as discussed in 3.2. Node with highest spread *v* is added to the set *S.* Then the algorithm starts the following local changes (Line 3) to current seed set *S* in attempt to improve influence spread.

i. **Delete** (Line 4) - If removing any node *v* in *S* increase the influence spread, then node *v* is removed from *S.*

ii. **Add** (Line 5) – If adding any node *v* not in *S* increase the influence spread, then node *v* is added to *S.*

iii. **Swap** (Line 6) – If removing any node *u* in *S* and adding a node *v* not in *S,* increases the influence spread, then node *u* is removed from *S* and node *v* is added to *S.*

Local search continues until any of the above local changes yields no further improvement.

4.2.1 Example

Now let us show how mineSeedLS() algorithm finds influential nodes as we discussed above using the an example social network graph in figure 2. Note that to compute influence spread under TGT model as discussed in section 3.2 we need to assign thresholds, θ^1 and θ^2, which are randomly assigned. For simplicity let us assume that for each node, threshold θ^1 and θ^2 is set to 0.3 and 0.6. Also let us assume that our *budget* is 2, that is, we are looking for 2 influential nodes.

First the algorithm will compute spread of each node as singleton. The node with maximum spread will be picked and stored into the set *S*. Table 6 below shows the spread of each node of graph in figure 2. As node u1 has the highest spread, it will be picked and added to the set of influential nodes *S*. So, at this point set *S* contains node u1 and its spread is 3. Now, the local search is going to start. The delete operation is skipped as there is only 1 node in the set *S*. Since our *budget* is 2 the algorithm continues to see if adding any node results in improving the spread. It picks node u2 as $\sigma_{TGT}(S + \{u2\}) = 4 > 3$ (note that $\sigma_{TGT}(S)$ was 3).

So, the set of influential set has two nodes {u1, u2}. Note that the spread of current set *S* is now 4. Then, it continues to check if swapping (or exchanging) any node in S with any node in *V* (but not already in *S*) yields any improvement in spread.

Algorithm 1 mineSeedLS
Input: Directed Graph *G*(*V,E*), *Budget*
Output: Set of influential nodes, S, such that |*S*| < Budget

---

1. *Set S to NULL*
2. *Compute spread of each node v ∈ V,* $\boldsymbol{\sigma_{TGT}(\{v\})}$*, and pick which yields highest spread and added to S.*
3. *Local Search on S to improve the selection by:*
4. ***Delete** node v, such that v ∈ S if*
   $\boldsymbol{\sigma_{TGT}(S-\{v\}) > \sigma_{TGT}(S)}$
5. ***Add** node v, such that v ∈ V-S, if |S| < Budget*
   $\boldsymbol{\sigma_{TGT}(S+\{v\}) > \sigma_{TGT}(S)}$
6. ***Swap** node u, u ∈ S with node v, v ∈ V-S if*
   $\boldsymbol{\sigma_{TGT}(S+\{v\}-\{u\}) > \sigma_{TGT}(S)}$
7. *Continue Step 3 while above local improvement in influence spread applies.*
8. *Return S*

Table 6: Spread of each nodes in figure 2.

Node *v*	u1	u2	u3	u4	u5
$\boldsymbol{\sigma_{TGT}(\{v\})}$	3	2	2	1	1

The spread of set *S* as a result of removing node u2 and replacing it with node u3 is 5, that is, $\sigma_{TGT}(S - \{u2\} + \{u3\}) = 5$. This is actually an improvement from previous spread of 4. So, node u2 is dropped and node u3 is added to set *S*, which now is {u1, u3}. The algorithm will continue to search for any further improvement. First, it checks if dropping any element from *S* improves the spread or not. Since $\sigma_{TGT}(S - \{u1\}) = 3$ and $\sigma_{TGT}(S - \{u2\}) = 2$ and does not improve the previous spread of 5, no element is dropped. Also the size of *S* is now 2 which is our budget, so the algorithm will not look for adding any new node. It will further check if swapping any node in *S* with any node in *V* (but not already in *S*) yields any improvement in spread. No exchange yields any improvement from pervious spread 5. So, the algorithm stops at this point and returns the set *S* = {u1, u3} which manages to achieve total gross spread of 5.

5. EXPERIMENTAL EVALUATION

5.1 Dataset

5.1.1 Epinions Dataset

Epinions.com is a product review site where users can read reviews about a variety of products or can join to begin writing reviews. Users of the Epinions.com can declare whether to "trust" or "distrust" each other. In this research we use Epinions dataset, provided by [12] and downloaded from http://www.trustlet.org that has two types of information. The first dataset consists of trust and distrust information. In this dataset we identified about 95, 318 nodes with 11,56,753 edges. However, since our main goal is to show that quality of nodes selected by mineSeedLS is better than that of greedy based solution under TGT model and also any network with nodes more than 10,000 may run for days, we select a small snap shot from the dataset comprising of approximately 10,000 nodes. The second dataset consists of rating information. Epinions users can post review on any certain

product. Other users can rate these reviews from 5 ("Very Helpful") to 1 ("Not Helpful") etc. We use this dataset to extract action log of users in the network. Whenever a user rates a review it is considered as an action performed by the users. Rating of each object is first classified as 'High' (if the rating is between 3-5) and 'Low' (if the rating is between 1-2). We consider two that users perform the same action if they rated the same object as 'High' or 'Low'.

5.1.2 Wikipedia Dataset

Wikipedia is a very popular free online encyclopaedia which is maintained and written by volunteers around the world. A small part of such contributors are 'administrators', who have access to additional technical features that help in maintenance. Users of Wikipedia can vote for or against another user to become administrator. The data set we collected from 'The Koblenz Network Collection' (http://konect.uni-koblenz.de/) consists of network of users from the English Wikipedia that voted for and against each other in admin elections. Nodes represent individual users of Wikipedia. And edges represent votes, which can be positive ("for" vote) and negative ("against" vote). In the dataset we have about 8,297 nodes and about 107,071 edges. Unfortunately there was no 'Action Log' available for this dataset. So we assigned influence probability uniformly and randomly to each edge.

5.2 Algorithms compared

The goal of our experiments is to show that influence spread achieved by our MineSeedLS algorithm improves influence spreads that can be achieved by standard approaches like CELF of [10]. We compared influence spread, number of nodes activated by seed set discovered, achieved by our proposed T-IM framework with the following approaches:

CELF-TGT: This is the greedy algorithm of [6] with the CELF optimization [10].

Degree-TGT: For comparison, we also compare our approach with a simple heuristic that selects the top k vertices with the highest degrees [6] [2]. Since we are dealing with trust network we select vertices with largest positive in degree.

5.3 Comparing Influence Spread

Figure 3 shows the influence spreads of various algorithms on trust network graph generated from Wikipedia dataset. Our T-IM performs very closely to CELF-TGT for smaller seed sets (<10). However, it outperforms CELF-TGT for seed set size > 15. It also outperforms Degree-TGT for all seed set sizes. Also in Epinions dataset the spread achieved by T-IM outperforms both Degree and CELF based solutions (Figure 4). As expected CELF performs inconsistently in both datasets and in some cases it even performs below the Degree based solution.

5.4 Comparing Run Time

To compare runtime of mineSeedLS with CELF and Degree based heuristic, we recorded time required to select influential nodes of different sizes. Figure 5 reports the runtime comparison on Epinions dataset. Degree heuristic performs almost in constant time. MineSeedLS takes longer than CELF as the size of the required set of influential nodes increases in both datasets. This shows the room for improvement of mineSeedLS in terms of scalability. As mentioned earlier, scalability was not focus of this work; however there are several ways to make the approach more scalable. We discuss some of these approaches in the next section.

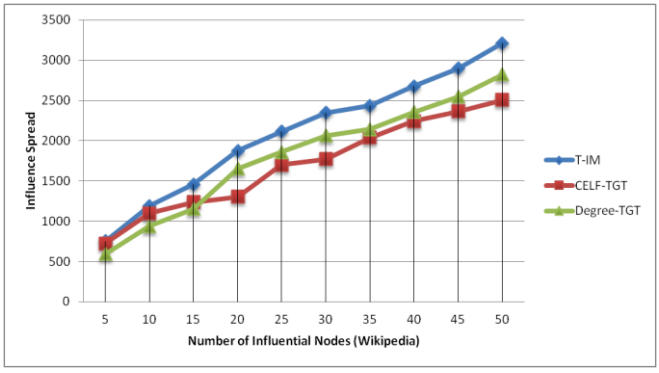

Figure 3: Influence spread of various algorithms in Wikipedia Dataset

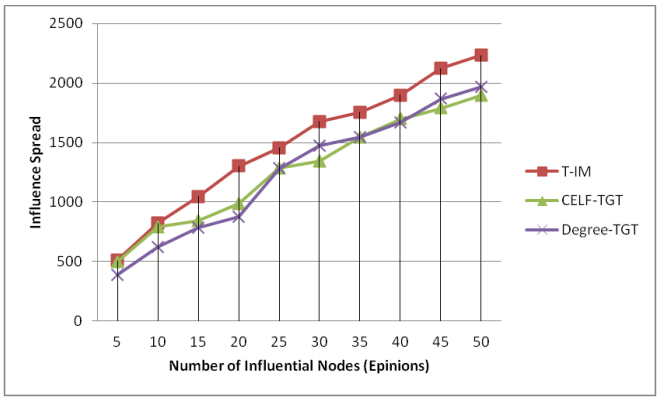

Figure 4: Influence spread of various algorithms in Epinions Dataset

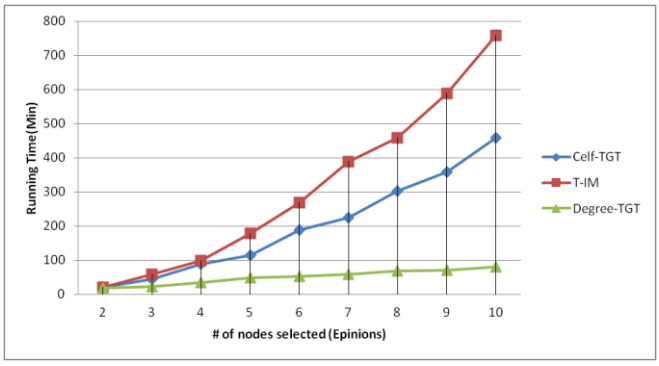

Figure 5: Running time of different algorithms on Epinions Dataset under TGT model

6. CONCLUSIONS

Analyzing information diffusion and social influence in social networks has various real-world applications. Influence maximization (IM) in viral marketing is an example of such an important application. In this research we tackled the influence maximization problem in trust network. We argue that to find influential nodes from a trust network we need to model the diffusion process by considering both positive and negative influence exerted by trusted and distrusted neighbours. Motivated by this, we introduce a new diffusion model, called Trust-General

Threshold (TGT) model, where both positive and negative influence exists. We showed that unlike existing diffusion models, influence maximization under proposed TGT model is a problem of *maximizing non-monotone sub-modular function*.

To learn influence probabilities, which are required parameters for TGT model, we mine action logs to extract frequent patterns of actions performed by users in trust network. Using these we estimate both positive and negative influence probabilities required for the TGT model. Then we propose an algorithm, called mineSeedLS, using local search technique [9] to find influential nodes. We ran experiments on real life dataset collected from Epinions and Wikipedia to show that quality of nodes selected by our proposed mineSeedLS outperforms existing benchmark algorithms such as CELF [10] by almost 35%.

However, as expected, the scalability of minedSeedLS is not suitable for large social network. Previously, scalability is tackled in Influence Maximization under various models such as LT and IC [2][3]. In the future we want to adopt some of these methods in our TGT model to make it more scalable.

7. REFERENCES

[1] Bonchi, F., Castillo, C., Gionis, A., and Jaimes, A. 2011. Social network analysis and mining for business applications. *ACM Trans. Intell. Syst*. Technol. 2, 3 (May), 22:1–22:37.

[2] Chen, W., Wang, Y., and Yang, S. 2009. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining.* KDD ’09. ACM, New York, NY, USA, 199–208.

[3] Chen, W. Wang, C. and Wang, Y. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (KDD '10). ACM, New York, NY, USA, 1029-1038.

[4] Domingos, P. and Richardson, M. 2001. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. KDD ’01*. ACM, New York, NY, USA, 57–66.

[5] Goyal, A., Bonchi, F., and Lakshmanan, L. V. 2010. Learning influence probabilities in social networks. In *Proceedings of the third ACM international conference on Web search and data mining. WSDM ’10*. ACM, New York, NY, USA, 241–250.

[6] Kempe, D., Kleinberg, J., and Tardos, E. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD ’03. ACM, New York, NY, USA, 137–146.

[7] Kimura, M. and Saito, K. 2006. Tractable models for information diffusion in social networks. In *Knowledge Discovery in Databases: PKDD 2006*. Lecture Notes in Computer Science, vol. 4213. Springer Berlin / Heidelberg, 259–271. 10.1007/11871637.

[8] Kleinberg, J. Cascading behavior in networks: algorithmic and economic issues. Cambridge University Press, 2007.

[9] Lee, J., Mirrokni, V. S., Nagarajan, V., and Sviridenko, M. 2009. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of the 41st annual ACM symposium on Theory of computing* (STOC '09). ACM, New York, NY, USA, 323-332.

[10] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N. 2007b. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD ’07. ACM, New York, NY, USA, 420–429.

[11] Richardson, M. and Domingos, P. 2002. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD ’02. ACM, New York, NY, USA, 61–70.

[12] Massa, P. and Avesani, P. Trust metrics in recommender systems. Massa, P. and Avesani, P. 2007. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems.* RecSys ’07. ACM, New York, NY, USA, 17–24.

[13] Nemhauser, G. L., Wolsey, L. A. and Fisher, M. L. An analysis of approximations for maximizing submodular set functions, *Mathematical Programming* 14, 1978, 265–294.