On Flow Authority Discovery in Social Networks

Charu C. Aggarwal[*] Arijit Khan[†] Xifeng Yan[‡]

Abstract

A central characteristic of social networks is that it facilitates rapid dissemination of information between large groups of individuals. This paper will examine the problem of determination of *information flow representatives*, a small group of authoritative representatives to whom the dissemination of a piece of information leads to the maximum spread. Clearly, information flow is affected by a number of different structural factors such as the node degree, connectivity, intensity of information flow interaction and the global structural behavior of the underlying network. We will propose a stochastic information flow model, and use it to determine the authoritative representatives in the underlying social network. We will first design an accurate *RankedReplace* algorithm, and then use a Bayes probabilistic model in order to approximate the effectiveness of this algorithm with the use of a fast algorithm. We will examine the results on a number of real social network data sets, and show that the method is more effective than state-of-the-art methods.

Keywords: Social networks, graph mining

1 Introduction

In recent years, social networks have found increasing popularity because of their ability to connect geographically disparate groups of individuals. Social networks are well known to enjoy the benefits of the *network effect*, wherein the increase in the size of the social network also increases the perceived benefits of using it. Much of this benefit is embedded in the information flows in the social network. These information flows arise as a result of the communication between the different entities in the social network. The information flow is also impacted by the network topology and the intensity of information flow interactions between different nodes. Since information flows play such a key role in the popularity of social networks, significant research has been performed in recent years to characterize important characteristics of such flows [13, 14, 22].

A key question which arises in the context of social networks is to determine the *information flow authorities* in the social network. Information flow authorities are defined as a very small group of members at which the dissemination of information leads to the most rapid spread throughout the social network. The concept of information authorities is peripherally related to that of the concept of *hubs and authorities* in web networks [10]. The concept of hubs and authorities is used in order to find central points of influence in web networks. However, the concept of *information flow authorities* is quite different from that of the hub-authority framework, in that it is more critically dependent upon the structure of the *flows along the underlying network*. This is dependent both upon the structural characteristics of the network and the flow intensity along different edges (which can be measured in many applications). In the experimental section, we will see that the use of a purely structural method (called *PeerInfluence*) is not sufficient for effective determination of flow authorities. The use of an *information flow model* is critical in determining the best nodes for information dissemination. Furthermore, our model also allows for the development of particular variants which can target specific nodes for influence. This is interesting in a number of applications in which only a subset of the nodes may be relevant for dissemination of information.

Clearly, the flow authorities in the social network are likely to be central and well connected entities in the network. This is related to the concept of determining *central nodes* [8] in graphs and social networks. However, the local structural measures alone do not provide a global view of the *centrality of flows* in the social network. Rather, the flow centrality is defined by the global topology, and the pattern of interactions between different nodes. A related problem is that of virus propagation in computer networks and epidemic spreading [4, 16, 19]. It has been observed in earlier work [4], that the flow of information in social networks, blogs, and network-based product-recommendation systems is very similar to that of virus spread in computer networks. It has been observed in this work that the structure of the network and the interaction intensities between nodes can play a critical role in the information dissemination process.

We will design a stochastic approach in order to

[*]IBM T.J. Watson Research Center, charu@us.ibm.com
[†]Univ. of California at Santa Barbara, arijitkhan@cs.ucsb.edu
[‡]University of California at Santa Barbara, xyan@cs.ucsb.edu

model the flow behavior in social networks. We will leverage this flow model in order to design an approach (called *RankedReplace*) for determining flow authorities in social networks. Then, we will approximate the flow model with a random-walk based model in conjunction with a probabilistic Bayes algorithm. We will refer to this algorithm as the *BayesTraceback* algorithm. This approximation is very efficient, and turns out to be almost equally effective in practice. We will show that our techniques are much more effective than state-of-the-art techniques which can be adapted to this problem.

1.1 Related Work The problem of epidemic spread in computer networks has been studied extensively in [4, 12, 11, 16, 19, 20, 22]. Much of this work studies patterns of information flows and conditions under which such flows becoming epidemics. Some recent research [22] studies the information propagation problem in context of similar models for computer virus and epidemic spreading [16, 19]. The work in [4, 18, 17, 22] studies the information flows in the context of social networks and other kinds of computer networks. However, none of these papers explore the concept of flow authorities in social networks. This problem is much more relevant in applications such as social networks, in which such transmission is desirable rather than undesirable. In this context, a method for mining the network value of customers in the context of a viral marketing model was proposed in [7]. While this body of work provides an excellent study of the flow behavior, it does not study the most influential points for optimal release.

Intuitively, flow authorities are likely to be present at highly connected and central regions in graphs. Centrality in graphs is typically defined either in terms of structural diameter or the betweenness centrality [1, 2]. Many algorithms have been designed for determining clusters and communities [1, 2] in massive graphs. Another such concept of centrality in the context of social networks is discussed in [8]. While such techniques provide a good *structural idea* of the main regions of the network, they fail to relate to the relationship between these regions and flows in the network.

The most closely related techniques to our work are discussed in [9, 5, 23], where an *independent cascade* (IC) model for information spread has been introduced. In the IC model, each active node gets a single chance to activate each of its neighbors independently with a certain probability. In the *Degree Discount* heuristic of the IC model [5], while selecting some node v as the authority node, we do not count the edge vu towards its degree, if u has already been selected as an authority node. However, this model only maximizes the expected number of nodes being exposed to the information, and it does not maximize the amount of the total flow of information in the network. Also, these techniques cannot be easily generalized to flows over specific portions of the network or targeting specific nodes. Our stochastic approach has a natural interpretation in terms of information flows in networks. As shown in the experimental section, this leads to the determination of much more relevant authorities. This paper will develop a Bayes model for constructing the flow with the use of a backward model known as the *BayesTraceback* algorithm. This approximation can provide an accurate determination of the flow authorities very efficiently.

2 Flow Authority Model for Social Networks

In this section, we will introduce the flow authority model for social networks. We assume that the universal set of nodes over which the social network is defined is denoted by U, and the edge set by A. Therefore, the underlying graph is denoted by (U, A). The graph is assumed to be directed, since information flows are specific to direction in the most general case. However, this assumption is not specific to the techniques discussed in this paper and they can easily be applied to undirected networks. This can be achieved by replacing an undirected edge with two symmetric directed edges. The set of nodes from which an incoming edge is incident into node i is denoted by $N(i)$. In other words, we have $N(i) = \{k : (k, i) \in A\}$. The set of nodes on which the outgoing edges of i are incident are denoted by $O(i)$. Therefore, we have $O(i) = \{k : (i, k) \in A\}$. We assume a model of information transmissibility, in which a node i which is *exposed to* information can transmit it to one of its neighbors. Information transmission can take on many forms in practical settings:

(1) In a social network, information may be forwarded to any of the friends of a given user in the form of publicly visible text posts, hyperlinks, videos or messages. This user may or may not choose to adopt this piece of information and transmit it further.

(2) In a peer-to-peer recommendation or viral marketing system, a user may send a recommendation to any neighbor. The neighbor may or may not make a buying decision based on this recommendation. Furthermore, this recommendation may be forwarded to one of the neighbors of the node. In general, it has been observed [7] that customers in a network-marketing system have a certain value in terms of their being able to influence other members of the network. The determination of flow authorities will help us in identifying key points in the network which lead to the greatest spread of information.

(3) The above dynamic is generally true for a variety of network-based epidemic outbreaks, and may be generalized to social networks, blog posts [14], water monitoring systems, or any general network infection system which has structural similarity to epidemic outbreaks [4].

We will formally define the concept of *information exposure.*

DEFINITION 1. *A node is said to be exposed to information bits $\mathcal{I}$, if at least one of its neighbors contains the information $\mathcal{I}$.*

It is important to note that the concept of neighborhood *information exposure* (as defined in this paper) only entails the *presence of the information* at one of its neighbors, rather than any further explanation of what is done with it. The default assumption is that if a node contains some information bits, then all of its neighbors are automatically exposed to those bits. The probability that such an exposure results in *eventual* information assimilation is determined by a transmission matrix, which we will discuss shortly. We denote the transmission probability along edge (i, j) by p_{ij}. Note that this transmission probability simply indicates the probability that an exposure of node i also results in the *information being assimilated by* node j. Node j then automatically becomes eligible to transmit to its neighbors. We denote the corresponding matrix of transmission probabilities by $P = [p_{ij}]$. We note that this matrix is extremely sparse, because it is often overlaid on very sparse graphs such as social networks. We note that if r_i be the probability that a given node i contains information $\mathcal{I}$, then it *eventually* transmits the information $\mathcal{I}$ to adjacent node j with probability $r_i \cdot p_{ij}$. The value of p_{ij} can often be estimated from the underlying data.

In this paper, we will examine the problem of picking a set of k points in the network which maximizes the aggregate probability of information assimilation over all nodes in the graph. We refer to these k nodes as the information authorities in the underlying network. We summarize the problem as follows:

PROBLEM 1. *Determine the set S of k data points at which release of the information bits $\mathcal{I}$ would maximize the expected number of nodes over which $\mathcal{I}$ is assimilated.*

We note that this is a particularly difficult problem, because the probability of the spread of the information at any particular node cannot be expressed easily in closed form. Rather, it is described in the form of a *non-linear system* of equations. We define $\pi(i)$ to be the steady-state probability that node i assimilates the information. Then, the expected steady state number of nodes which assimilate the information are given by $\sum_{i \in U} \pi(i)$. In order for node i to assimilate the information, it must receive the transmission from *at least* one of its neighbors. The flip side of this argument is that in order for node i to not assimilate the information, it must not receive the transmission from *any* of its neighbors. The probability that none of the neighbors of node i transmit to it is given by $\prod_{l \in N(i)} (1 - \pi(l) \cdot p_{li})$. Therefore, we have:

$$1 - \pi(i) = \prod_{l \in N(i)} (1 - \pi(l) \cdot p_{li}) \tag{2.1}$$

In addition, for each of the k nodes in S at which the information is released, we set the corresponding value of $\pi(\cdot)$ to 1. Therefore, we have:

$$\pi(i) = 1 \quad \forall i \in S \tag{2.2}$$

The above system of equations is nonlinear, since it uses a product of the probability of (non-)exposure from different neighbors. This is a difficult set of equations to solve, and the corresponding result can only be obtained via numerical estimation. Furthermore, it is required to determine the set S optimally. The optimization problem is even more challenging. We will now restate Problem 1 more formally in terms of the relationships discussed above:

DEFINITION 2. *Determine the set S of nodes which maximizes $\sum_{i \in U} \pi(i)$ subject to the following constraints:*

- $1 - \pi(i) = \prod_{l \in N(i)} (1 - \pi(l) \cdot p_{li}) \quad \forall i \notin S$
- $\pi(i) = 1 \quad \forall i \in S$

Next, we will describe a simple algorithm to determine the information authorities with the use of an iterative numerical method. Later, we will present a much faster probabilistic method for the same problem. This method uses a Bayes model in order to determine the optimal flow authorities.

3 Determining Optimal Information Flow Authorities

In this section, we will present algorithms for determining optimal information flow authorities. In order to determine optimal flow authorities, we also need to have a way to evaluate the (aggregate) steady state assimilation probability of all nodes, when the information is released at a *particular* set of nodes S. In order to design this algorithm, we will use an iterative algorithm in which $q^t(i)$ denotes the estimation of $\pi(i)$ in the tth iteration. This iterative approach is natural to solve the

```
Algorithm SteadyStateSpread(Initial Set: $S$,
                Transmission Matrix: $P$)
begin
  for each $i \in S$ set $q^0(i) = 1$;
  for each $i \notin S$ set $q^0(i) = 0$;
  $t = 0$;
  repeat
    for each $i \in S$ set $q^{t+1}(i) = 1$;
    for each $i \notin S$ do
    begin
      $q^{t+1}(i) = 1 - \prod_{l \in N(i)} (1 - p_{li} \cdot q^t(i))$;
    end
    $C_{t+1} = \sum_{i \notin S} |q^{t+1}(i) - q^t(i)|$;
    $t = t + 1$;
  until($C_t < 0.01 \cdot C_1$);
  return( $\sum_{i \notin S} q^t(i)$);
end
```

Figure 1: Determining the expected information spread for a given starting set of nodes

non-linear system of equations. In each iteration, we update the value of $q^t(i)$ from the value of $q^{t-1}(i)$ with the use of the equations in Definition 2. The overall algorithm is denoted by *SteadyStateSpread* in Figure 1. The input to the algorithm is the set S at which the information is released.

The algorithm initializes $q^0(i) = 1$ for each node $i \in S$ and 0 for nodes which are not in S. Subsequently, an iterative approach is used to update the value of $q^{t+1}(\cdot)$ is updated from $q^t(\cdot)$ with the use of the equations in Definition 2. In each iteration, we track C_t, which is the aggregate change in the absolute probabilities from $q_t(\cdot)$ to $q_{t+1}(\cdot)$. The algorithm is terminated when the change in a given iteration is less than 1% of the change in the first iteration. At this point, it is assumed that the probability values have converged to values which are close to their true values.

The above method for determining the steady-state probabilities can be leveraged in order to determine the optimum set of k nodes at which the information should be released. We make use of a greedy approach which maximizes the expected increase in the information spread as calculated by Figure 1. The algorithm works with the use of an iterative approach in which we start off with a candidate set of k nodes and continually increase its maximum flow value. We first pick the top k nodes with the largest individual steady state spread as the initial candidate set of flow authorities. Of course this way of picking the candidates ignores the structural relationships between these nodes. In general, we would like our flow authorities to be reasonably well separated from one another in order to maximize the probability of propagation of information throughout the social network. In order to achieve this goal, we use a *ranked replace* algorithm in which the nodes in $U - S$ are tried as possible replacements for nodes in S in decreasing order of their flow value.

```
Algorithm RankedReplace(Transmission
         Matrix: $P$, NumberOfAuthorities: $k$);
begin
  Determine $SteadyStateSpread(\{i\}, P)$ for
   each node $i$ in the universal set $U$;
  $S$ =Initial set of $k$ authority nodes with the
     highest value of $SteadyStateSpread(\{i\}, P)$;
     Sort nodes in $U - S$ in descending order of
      $SteadyStateSpread(\cdot)$;
  for each node $i$ in $U - S$ in
       descending order do
  begin
    Sort the list $S$ in ascending order
      of $SteadyStateSpread(\{j\}, P)$;
    Pick the first element (if it exists) of
     sorted list $S$ which is such that
     replacing $i$ with it increases value of
     $SteadyStateSpread(S, P)$
    if no replacement has occurred in the last
    $r$ consecutive iterations, then
      return($S$) and terminate;
  end
  return($S$);
end
```

Figure 2: The *RankedReplace* Algorithm

The iterative portion of the algorithm proceeds as follows. We sort the nodes in $U - S$ in descending order of the steady state flow. In each iteration, we pick the next node i from $U - S$ and use it to replace a node in S, if such a replacement increases the total flow of S. Even though the flow value of i is typically less than that of the node it replaces, the total flow value may increase because of the nature of the network location of the two nodes. For this purpose, the nodes in S are tried as candidates for replacement in ascending value of $SteadyStateSpread(\cdot)$. The first replacement in this order which increases the objective function for the steady state information spread is executed. It is possible that no such replacement may exist. We continue to try different nodes in $U - S$ for replacement, until such attempts are unsuccessful for r consecutive iterations. At this point, the algorithm terminates, and the set of nodes S are reported as the flow authorities. The overall algorithm is illustrated in Figure 2. The algorithm is referred to as *RankedReplace*, which corresponds to the broad approach of ranking the nodes and iterative replacement based on the steady state flow impact.

3.1 The *BayesTraceback* Algorithm The main problem with the solutions presented in the previous sections is that the algorithms require iterative determination of the steady-state probabilities. This can be rather slow in practice. In this subsection, we will discuss how to speed up the algorithms for determination of the information authorities. This algorithm provides an an approximation of the information authorities. The core-idea is to use a *random walk* based approach in which an information packet is viewed as a token, and it is assumed that the token at a given node j is inherited from one of its *incoming nodes* i with probability proportional to p_{ij}. Random walk modeling is used for the page rank problem, though this approach is different in the sense that we use it for *trace back* of the best *source of* information, rather than those nodes which will be visited often by a random surfer. Thus, the algorithm tends to be *backward looking from a desired result*, rather than *forward looking to determine the result.* In the experimental section, we will show that a direct application of the page rank model does not yield as accurate results as the *BayesTraceback* method.

The random walk model is a relaxation of the original model for two reasons:
(1) In the original model, a node can be infected only once, whereas a random walk can visit a node multiple times.
(2) In the original model, a given node may infect multiple nodes at once, whereas in this case, we are trying to trace the behavior of a single token, which is (stochastically) present only at one node at a time.

We note that this simplification of the model allows us a trace-back of the steady-state probabilities with the use of a Bayes model. We will see that this approach is extremely efficient and provides a good approximation to the exact algorithm.

In the case of the random walk model, our aim is to pick k nodes in the data which are such that by releasing the information at these k points, the information spreads as evenly over the entire network as possible. Intuitively, this corresponds to release points which results in as much of the network being disseminated with the information as possible. We note that the even spread of information may not be possible in steady-state, since the *steady-state* probabilities in a random-walk model are dependent upon the structure of the network and the transition probabilities, and are independent of the initial starting point probabilities. Nevertheless, our goal is to create an evenly spread probability distribution as an *intermediate transient* after a small number of iterations of the walk model. The goal is to find a set of k starting points which will create such an intermediate transient at some point. Therefore, for a network

```
Algorithm BayesTraceback(Transmission Matrix: P
        Discard Fraction: f, NumberOfAuthorities: k);
begin
  t = 0;
  for each node i set $q^0(i) = 1/n$;
  repeat
    $q^{-(t+1)}(j) = \sum_i q^{-t}(i) \cdot \frac{p_{ji}}{\sum_{l \in N(i)} p_{li}}$
    t = t + 1;
    Remove a fraction f of the nodes
      from the graph with the least value of
      $q^{-(t+1)}(\cdot)$, with the restriction
        that at least k nodes should remain;
    Scale up probabilities $q^{(-(t+1)}(\cdot)$ of
      all remaining nodes by the same factor
      so that the remaining probabilities sum to 1;
  until(k nodes remain);
  return remaining nodes;
end
```

Figure 3: The *BayesTraceback* Algorithm

containing n nodes, we will start off with a *final transient probability distribution* of $1/n$ for each node, and then use the Bayes theorem repeatedly to *trace back* the initial probabilities for a certain number of iterations, and pick the k nodes with the largest apriori probability with the use of this traceback technique. Therefore, we start off with the probabilities for n nodes which are denoted by $\overline{q^0(\cdot)} = q^0(1) \ldots q^0(n)$. As noted earlier, each of these values is equal to $1/n$. In subsequent iterations, we will use the Bayes formula to determine the values of $\overline{q^{-1}(\cdot)}, \overline{q^{-2}(\cdot)} \ldots \overline{q^{-r}(\cdot)}$. Note that we use negative superscripts for the time component in order to denote the traceback starting from the 0th step of the walk. The vector $\overline{q^{-t}(\cdot)}$ indicates the probabilities after tracing the walk back for t steps.

Next, we will examine how the values of $\overline{q^{-(t+1)}(\cdot)}$ can be determined from $\overline{q^{-t}(\cdot)}$. For any particular node i, let us examine all the incoming edges from the corresponding node set $N(i)$. We note that the a-priori probability $P(j \to i | -t\text{th node} = i)$ that an information token at node i came from node j in the previous step of the random walk is given by the Bayes formula over all possible nodes incoming into node i. Therefore, we have:

$$P(j \to i | -t\text{th node} = i) = \frac{p_{ji}}{\sum_{l \in N(i)} p_{li}} \qquad (3.3)$$

In order to trace back the values of $\overline{q^{-(t+1)}(\cdot)}$ from $\overline{q^{-t}(\cdot)}$, we can examine the different cases over which the $-t$th node is i and sum up the values of $P(j \to$

$i| - t$th node $= i$) over these cases. Therefore, we have:

$$q^{(-(t+1))}(j) = \sum_i q^{-t}(i) \cdot P(j \to i| - t\text{th node} = i)$$
$$= \sum_i q^{-t}(i) \cdot \frac{p_{ji}}{\sum_{l \in N(i)} p_{li}}$$

The second equation above simply traces back for the probability distribution of the position of the information token at time stamp $-(t+1)$ using the probability distribution of the token at time stamp $-t$. Therefore, we can start off with the evenly distributed probability vector $q^0(\cdot)$ and start tracing back the probabilities. The nature of the above probabilities suggest that nodes with high outdegree and outgoing probabilities will see increased probability during the traceback process. The process above is repeated for r iterations, and then the k nodes with the largest value of $q^{-r}(i)$ are picked as the correct candidates. It remains to describe the termination criterion. Furthermore, we need to design the algorithm in such a way, so that the algorithm converges.

It turns out that both of the above issues can be solved by making a heuristic change to the algorithm. This heuristic change speeds up the convergence and also provides a natural termination criterion to the algorithm. Note that since we only wish to determine the high probability nodes after the traceback, we can start removing the nodes, whose influence to this is minimal. After each iteration of updating $q^{-(t+1)}(\cdot)$ from $q^{-t}(\cdot)$, we conceptually discard a fraction f of nodes with the least probability (least value of $q^{-(t+1)}(\cdot)$), by setting the corresponding values of $q^{-(t+1)}(\cdot)$ to zero. We also delete the corresponding nodes and edges from the graph. At the same time, we scale up the probabilities of the remaining nodes (by the same factor), so that they continue to sum to 1. This process is repeated iteratively until exactly k nodes are remaining. Note that the last iteration is special in the sense that less than a fraction f of the nodes may need to be dropped in order to ensure that we continue to have k nodes remaining. These k nodes are reported as the information authorities. The overall algorithm is illustrated in Figure 3. The input to the algorithm is the *discard fraction* f and the transition matrix P. The choice of the *discard fraction* f determines the speed of termination of the algorithm. A larger choice of f leads to faster convergence, but somewhat more inaccurate results. In practice, we chose f to be about 5% of the total number of nodes. We note that this algorithm is extremely efficient, since each iteration is a straightforward update step on the different nodes. Furthermore, for a graph containing n nodes, the maximum number of iterations is $\log(n/k)/\log(1/(1-f))$. This is because the number of nodes reduces by a factor of $(1-f)$ in each iteration, and the number of nodes need to be reduced from n to k in all iterations. Because of the logarithmic variation, this turns out to be quite modest. For example, for a network containing 10^6 nodes, $k = 10$ and $f = 0.05$, the total number of iterations is less than 180.

We note that successive removal of nodes and edges from the graph will eventually lead to the underlying graph becoming disconnected. This does not change the overall algorithm, since the iterative transition relationships continue to hold within each connected component. Conceptually, the algorithm will eventually find the "most significant nodes" in the k highest probability components.

3.2 Restricting Source and Target Nodes In the previous discussion, we picked the most relevant flow authorities for the entire set of nodes. In this section, we will examine the case when we wish to determine the flow authorities for a particular set T of target nodes. Such situations may arise in a number of scenarios in which a user may target a particular subset of nodes on which the information flow needs to be maximized. This problem can be achieved by simple modifications to each of the above algorithms:
(1) For the case of the *RankedReplace* algorithm, the only change is to modify the *SteadyStateSpread* algorithm. In the modified algorithm, we add the set T to the input parameters. The actual state probabilities on the nodes are computed using the same algorithm as before, except that the final information flow value which is returned is determined by summing up these probability only over T rather than the entire set of nodes. When the *RankedReplace* algorithm is executed with this new method of determining the steady state flow, it automatically picks the set of flow authorities which maximize the flow to the target set T.
(2) For the case of the *BayesTraceback* algorithm, we consider the nodes within target set as the sink nodes. Note that, the nodes that have the maximum influence over a set of target nodes intuitively correspond to the nodes that can evenly spread the information within the target nodes as quickly as possible. However, to achieve the maximum flow within target nodes, we are free to take help of non-target nodes. Now, in this modified *BayesTraceback* approach, the algorithm still remains the same inside the subgraph imposed by the target nodes. For the subgraph imposed by the non-target nodes, we do not care the total flow that could be aggregated there by the whole process, as this is used only for the flow propagation within the target nodes. Therefore, the only change to the method is that we do not propagate the flow from target node to non-target node, but we propagate flow from non-target to target sets.

It is further possible to restrict the set of influential nodes to a particular set S. This situation can arise in cases, where the information can be released only at specific nodes. This generalization can be solved by adding this as an input parameter in case of *RankedReplace* algorithm. We only use the nodes in S for the ranking process in this case. For the case of the *BayesTraceback* algorithm, we run the algorithm in the same way as the previous case, except that the top-k nodes from the set S are picked as the final solution.

4 Experimental Results

We will present experimental results which illustrate the effectiveness, efficiency and robustness of our techniques on a number of real data sets. To compare our results, we consider some of the structural and random walk based algorithms as natural baselines. For example, we implemented the *Recursive Neighbor Mean (RNM) Algorithm* [15], which determines the peer influence groups and thereby identifies the dense clusters in a large network. The node with the highest degree centrality [6] in each cluster is considered the authority node in each cluster using this baseline approach. We refer to this algorithm as *Peer-Influence* in the experimental section. We also implemented the *Degree Discount IC* heuristic [5] discussed earlier. In the IC model, each active node gets a single chance to activate each of its neighbors independently with a certain probability. In the *Degree Discount* heuristic of the IC model, while selecting some node v as the authority node, we do not count the edge vu towards its degree, if u has already been selected as an authority node. Finally, we compare our top-k flow authority nodes with the top-k nodes having the highest *PageRank* [24] values.

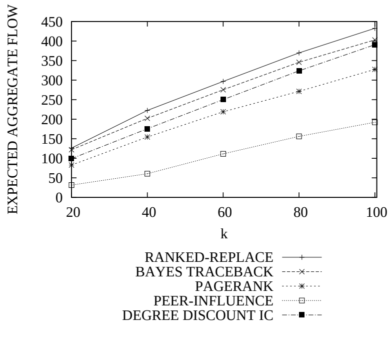

Figure 4: Effectiveness Results (*DBLP*)

4.1 Data Sets

The algorithms were tested on a variety of different kinds of interaction networks. These interaction networks were constructed from a number of different kinds of social network settings. We describe the data sets [1] in detail below.

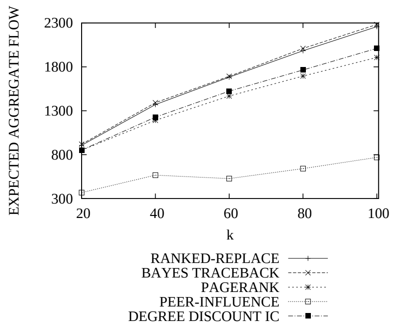

Figure 5: Effectiveness Results(*last.fm*)

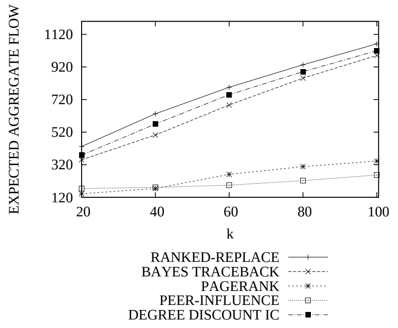

Figure 6: Effectiveness Results (*Twitter*)

DBLP Collaboration Network: We use the well known DBLP collaboration graph [2] consisting of 684, 911 distinct authors and 7, 764, 604 collaboration edges among them. We define the transmission probability of an edge to be proportional to the number of times that the two authors publish a paper together. The proportionality factor is the inverse of the maximum number of collaborations between any pair of authors in the network.

Last.fm Social Network: We crawled a social network consisting of 818, 800 users from the *last.fm* site. This is a music web site where users listen to their favorite tracks and communicate with each other based on their choice of music. There are a total of 3, 340, 954 edges among these users. In each case, an edge represents user posts which correspond to song recommendations between users. The transmission probability of an edge is proportional to the number of times a recommendation was sent from one user to another. The proportionality factor is the inverse of the maximum number of communications between any two users.

Twitter Social Network: We crawled a so-

[1]Our data and codes are available in http://cs.ucsb.edu/∼arijitkhan/Gflow.zip

[2]http://www.informatik.uni-trier.de/ ley/db/

Rank	*Ranked-Replace*	*Bayes Traceback*	*Peer-Influence*	*Degree Discount IC*
1	Wen Gao	Wen Gao	Luigi Fortuna	Wei Li
2	Francky Catthor	Philip S. Yu	Dipanwita Roy Chowdhury	Wei Wang
3	Philip S. Yu	Mahmut T. Kandemir	Timothy D. Sullivan	Li Zhang
4	Mahmut T. Kandemir	Francky Catthoor	Wei Li	Ian T. Foster
5	A. L. S. Vincentelli	A. L. S. Vincentelli	S. C. Lin	Wei Zhang
6	Elisa Bertino	Thomas S. Huang	E. K. Zavadskas	Ming Li
7	Thomas S. Huang	Elisa Bertino	Kellie J. Archer	Lei Zhang
8	Ian T. Foster	Wei-Ying Ma	Herman Van Keulen	Lei Wang
9	Luca Benini	Donald F. Towsley	Weichung Wang	A. L. S. Vincentelli
10	Hans-Peter Seidel	Ian T. Foster	Roman Andrushkiw	Jun Wang
11	Wei-Ying Ma	Erik D. Demaine	Apinunt Thanachayanont	Wei Liu
12	Erik D. Demaine	Hans-Peter Seidel	Horst Zimmermann	Jun Zhang
13	Ming Li	Ming Li	S. P. Perone	Jing Wang
14	Donald F. Towsley	Van Keulen	Cndido Lpez-Garca	Li Li
15	Wei Wang	Jiawei Han	M. McCormick	Francky Catthoor
16	Wei Li	HongJiang Zhang	Chao Jiang	Ying Zhang
17	Mario Piattini	P. Nagley	Leon F. Osborne	Elisa Bertino
18	Hsinchun Chen	Joel H. Saltz	Jack Dongarra	Xin Li
19	Li Zhang	Mary Jane Irwin	John P. Woodruff	Wen Gao
20	Hector Garcia-Molina	Gerhard Weikum	Aarne Halme	Hui Zhang

Table 1: Examples of Results Obtained by Different Methods

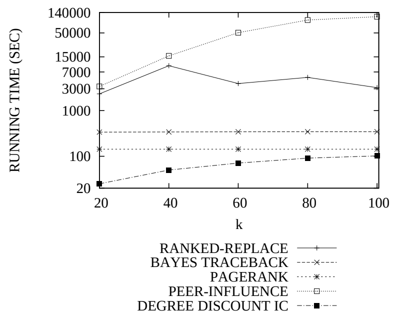

Figure 7: Efficiency Results (*DBLP*)

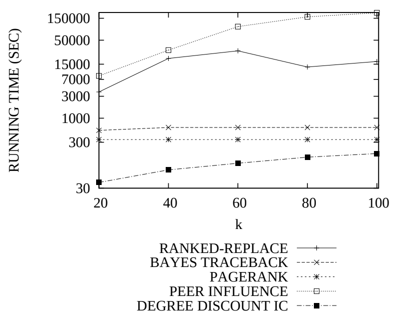

Figure 8: Efficiency Results (*last.fm*)

cial network consisting of $1,994,092$ users from http://twitter.com. Twitter is a free social networking and micro-blogging service that enables its users to send and read messages There are a total of $6,450,193$ edges among these users. In each case, an edge represents messages sent from one user to another. The transmission probability of an edge is proportional to the number of times the users have communicated. As in the previous case, the proportionality factor is the inverse of the maximum number of communications between any pair of users.

4.2 Case Studies Before more concrete presentation of the effectiveness results with quantitative measures on the information spread, we will provide an intuitive exploration of the results obtained with the different algorithms for the **DBLP** data set. This provides an intuitive understanding of the nature of the results obtained by the different methods. We provide the name of authority nodes for $k = 20$ in Table 1. It is evident that the authority nodes determined by the *Ranked-Replace* and *BayesTraceback* algorithms mostly contain well-known and influential researchers from different fields of computer science. Furthermore, we note that even though the algorithms are quite different from one another, the authority nodes determined are quite similar. Furthermore, these researchers are *structurally placed* in such a way so as to maximize the interaction with other researchers. All these factors contribute to the total aggregate flow across the entire network. The *Peer-Influence* method is particularly poor in determining good authority nodes, because it does not properly

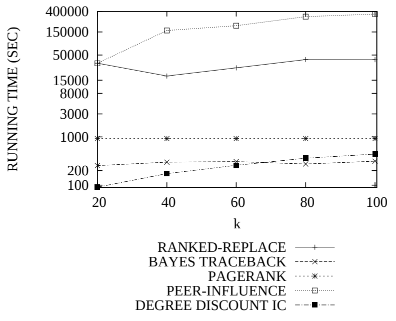

Figure 9: Efficiency Results (*Twitter*)

compute the flows on the basis of *random walk behavior*, and pure structural diameters simply do not encode enough information to ensure robustness. This results in lower aggregate flow across the whole graph. We also tested the *Degree Discount IC* algorithm. The *Degree Discount IC* algorithm determines better quality results than the *Peer-Influence* algorithm, because it uses a weighted version of random-walk, where the weight is determined by the degree of a node and the corresponding transmission probabilities; however, the determined authority nodes are quite different from the *RankedReplace* and *BayesTraceback* algorithms, because it performs a forward calculation as opposed to Bayes-based backward measures. This difference is quite significant; in the next section, we will use quantitative measures on the information spread to show that the *RankedReplace* and *BayesTraceback* algorithms are more effective than the *Degree Discount IC* algorithm in many cases.

4.3 Effectiveness Results In order to measure the effectiveness of a set of authority nodes S, we used expected value of the steady state flow from the determined set S to the remaining set of nodes. We determine the expected aggregate flow for different values of k, the number of authority nodes. Figure 4 illustrates the effectiveness result for the **DBLP** data set. The value of k is illustrated on the X-axis, whereas the flow value is illustrated on the Y-axis. The expected aggregate flow increases with the number of authority nodes, since the release of information at a larger number of nodes leads to greater spread of information. The *Ranked-Replace* method slightly outperforms the *BayesTraceback* algorithm. We will see that the *BayesTraceback* method is also extremely efficient, and therefore it is the most practical alternative among the different methods. Furthermore, both techniques perform *significantly* better than the three baseline techniques. For example, when we set $k = 60$, the expected aggregate flow using the *Ranked-Replace*, *BayesTraceback*, *PageRank*, *Degree Discount IC* and the *Peer-Influence* methods are 296.70, 275.42, 211.67, 250.28 and 111.48 respectively.

Figure 5 illustrates the effectiveness results of our method for the **last.fm** data set. Both the *Ranked-Replace* and *BayesTraceback* algorithms perform very similarly, and also significantly outperform the three baseline methods. For $k = 60$, the expected aggregate flow using the *Ranked-Replace*, *BayesTraceback*, *PageRank*, *Degree Discount IC* and the *Peer-Influence* methods are 1682.62, 1692.21, 1450.62, 1523.50 and 527.27 respectively. In Figure 6, we illustrate the results for the **Twitter** data set. In this case, the *Degree Discount IC* heuristic performs slightly better than the *BayesTraceback* method. For example, for $k = 80$, the expected aggregate flow using the *RankedReplace*, *BayesTraceback*, *PageRank*, *Degree Discount IC* and the *Peer-Influence* methods are 933.64, 851.47, 258.76, 891.24 and 222.34 respectively.

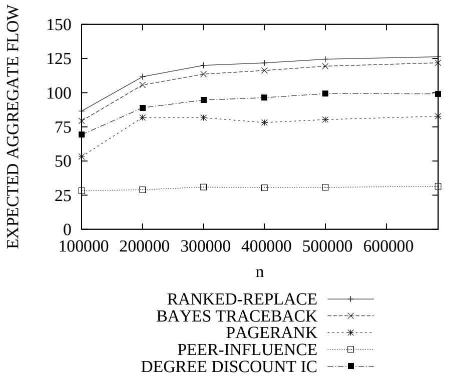

Figure 10: Effectiveness vs. Network Size (*DBLP*)

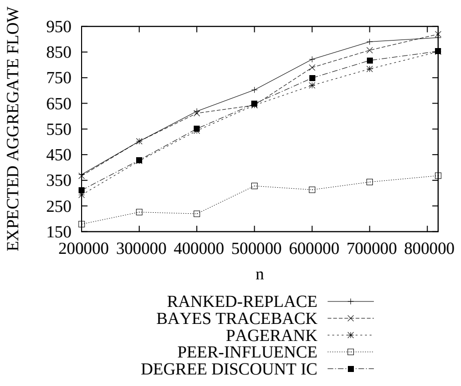

Figure 11: Effectiveness vs. Network Size (*last.fm*)

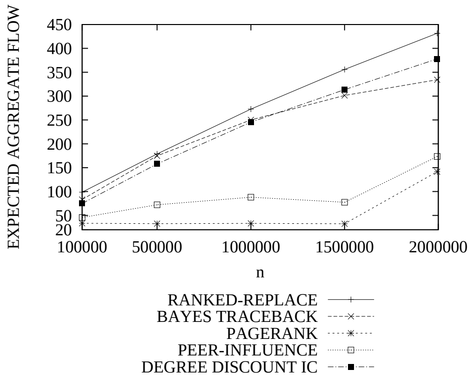

Figure 12: Effectiveness vs. Network Size (*Twitter*)

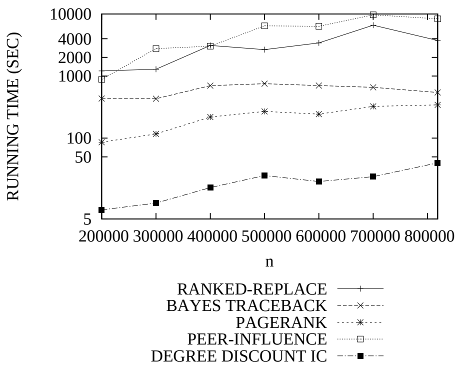

Figure 14: Efficiency vs. Network Size (*last.fm*)

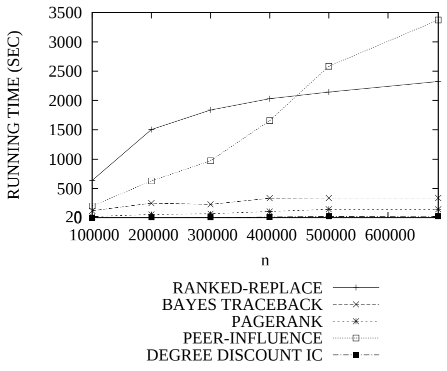

Figure 13: Efficiency vs. Network Size (*DBLP*)

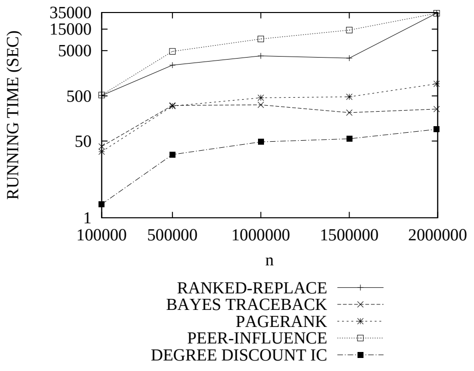

Figure 15: Efficiency vs. Network Size (*Twitter*)

4.4 Efficiency Results We compare the running time of the *RankedReplace* and *BayesTraceback* methods with that of the three baseline methods. Figure 7 shows the efficiency result for the **DBLP** data set. The number of authority nodes k is varied from 20 to 100 on the X-axis, whereas the running time is illustrated on the Y-axis. The *Peer-Influence* approach is the most inefficient, and its running time increases rapidly with the number of authority nodes. The *BayesTraceback* algorithm, on the other hand, is very efficient, though the *Degree Discount IC* algorithm is the fastest. For $k = 60$, the running time of the *RankedReplace*, *BayesTraceback*, *PageRank*, *Degree Discount IC* and *Peer-Influence* methods are 3904, 343, 104, 71 and 50743 seconds respectively. The running times for the **last.fm** data set are illustrated in Figure 8. For $k = 60$, the running time of the *Ranked-Replace*, *BayesTraceback*, *PageRank*, *Degree Discount IC* and the *Peer-Influence* method are 29265, 628, 301, 105 and 98930 seconds respectively.Thus, the *Peer-Influence* method is two orders of magnitude slower than our two methods. Also, the *BayesTraceback* method is an effective alternative to the *RankedReplace* method, while maintaining a significantly high level of effectiveness.

Besides, the time requirement for the *BayesTraceback* method does not vary much with respect to k. This is because a fixed fraction of the nodes are discarded in each iteration, and the number of iterations for convergence of this method is inversely proportional to $\log n$. The *Ranked-Replace* method is a greedy approach and it does not follow any specific pattern with respect to k. However, the running time of the *Degree Discount IC* is proportional to k [5]. We note that, in the **Twitter** dataset (Figure 9), the *BayesTraceback* approach is more efficient than the *Degree Discount IC* heuristic for higher values of k. For example, when we set $k = 80$, the running times of the *Ranked-Replace*, *BayesTraceback*, *PageRank*, *Degree Discount IC* and the *Peer-Influence* methods are 40225, 275, 1001, 362 and 312345 seconds respectively.

4.5 Robustness and Scalability with increasing Network Size Our goal in this section is to test the robustness and scalability of the method with increasing network size. This will show the effectiveness of the

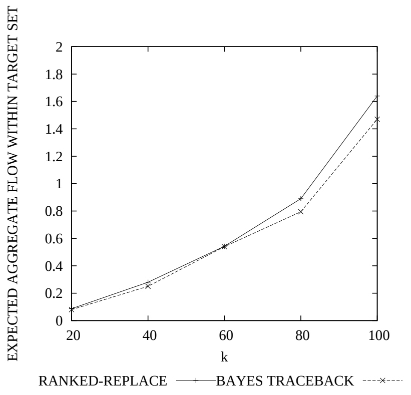

Figure 16: Maximum Aggregate Flow for Particular Target Nodes (*DBLP*)

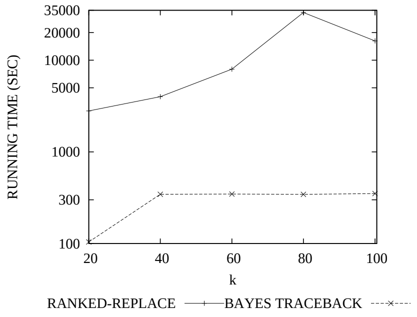

Figure 17: Running Time to Find Authority Set for Particular Target Nodes (*DBLP*)

method with different network sizes, and also the scalability of the method. In order to obtain networks of increasing sizes, we randomly deleted nodes (and their incident edges from the data set), and tested the algorithm over networks of increasing size. We set k, the number of authority nodes, as 20. We provide results on the effectiveness and efficiency for increasing number of nodes.

Figure 10 shows the variation of the maximum information spread with increasing number of nodes for the **DBLP** data set. The total flow value initially increases with the number of nodes, because the full benefit of multiple points of information release in small networks is not realized. On the other hand, if the networks are too large, then the information spread may get sufficiently damped in a few iterations. Therefore, the flow value increases relatively fast up to the value of $n = 300,000$ for both the *Ranked-Replace* and *BayesTraceback* methods, and then levels off. The *Ranked-Replace* and *BayesTraceback* method both outperform the baseline approaches by a high margin for all values of n. This suggests that the method is extremely robust over networks of different sizes. Figure 11 shows the corresponding results for **last.fm** data set. As in the case of the **DBLP** data set, the expected aggregate flow for the *Ranked-Replace* and *BayesTraceback* methods are much higher than that of *Degree Discount IC*, *PageRank* and the *Peer-Influence* methods over the entire range of possible network sizes. We plot the expected aggregate flow for different network sizes for the case of the **Twitter** data set in Figure 12. The results are similar to the case of the other two data sets when the number of nodes is less than $1,000,000$. However, for n greater than $1,000,000$, the *Degree Discount IC* heuristic performs slightly better than the *BayesTraceback* method.

Figure 13 illustrates the running time scalability with increasing number of nodes for the **DBLP** data set. For the *BayesTraceback* method, the running time does not vary much with respect to the number of nodes since the number of iterations increases only logarithmically with the number of nodes. As observed earlier, it is slower than the *Degree Discount IC* method but significantly faster than either the *RankedReplace* or the *Peer-Influence* methods. For small values of the number of nodes n, the *Peer-Influence* approach is slightly faster than the *Ranked-Replace* method; however, the former does not scale well and is much slower than the *Ranked-Replace* method for larger networks. For $n = 500,000$ or higher, the *Peer-Influence* approach requires significantly more time than the *Ranked-Replace* technique. The results for the **last.fm** data set show similar trends, as is evident from Figure 14. The results for the **Twitter** data set are illustrated in Figure 15. In this case, the running times of *Degree Discount IC* and *BayesTraceback* algorithms are comparable. These algorithms are also significantly faster than the other two methods.

4.6 Targeted Flow Authorities

We also tested our two schemes for the case when we determined the flow authorities for a particular set of target nodes. For the **DBLP** collaboration graph, we randomly selected a set of 1000 target nodes and determine the corresponding authority nodes which will maximize the flow within that target set. Figure 16 illustrates the expected aggregate information spread within this target set (for different number k of authority nodes) using the modified *Ranked-Replace* and *BayesTraceback* methods. Figure 17 illustrates the corresponding running time to find the authority nodes. In this case, we do not show the baseline methods because they cannot be easily modified when particular nodes are targeted. Thus, our scheme also provides better functionality than the baseline methods. It is evident that the modified *BayesTraceback* method performs almost as well as the modified *Ranked-Replace* technique; however it is significantly faster in terms of running time. Unlike the *RankedReplace* method, the running time of the *BayesTraceback* method is relatively insensitive to the number of authority nodes k. Therefore, the *BayesTraceback* method provides the best tradeoffs between quality and efficiency.

5 Conclusions and Summary

In this paper, we designed an algorithm for the determination of optimal flow authorities in social networks. We designed two algorithms for the task, which correspond to the *RankedReplace* and *BayesTraceback* algorithms. We presented experimental results illustrating the effectiveness of our methods on a number of social networking and collaboration graphs. Our results show that the techniques proposed in this paper are much more effective than the currently available techniques. While the *RankedReplace* technique is slightly more effective than the *BayesTraceback* method, the latter is significantly more efficient. Furthermore, it is much superior to the baseline methods in terms of effectiveness. The *BayesTraceback* algorithm provides the best tradeoff between quality and efficiency.

Acknowledgements

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053, and in part by the Institute for Collaborative Biotechnologies through Grant No. DFR3A-8-447850-23002 from the U.S. Army Research Office. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] C. Aggarwal, and H. Wang, Managing and Mining Graph Data, *Springer*, (2010).

[2] C. Aggarwal, Social Network Data Analytics, *Springer*, (2011).

[3] N. Berger, C. Borgs, J. T. Chayes, and A. Saberi, *On the spread of viruses in the internet*, SODA, (2005).

[4] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos, *Epidemic thresholds in real networks*, ACM Trans. on Inf. Systems and Security, 10(4), (2008).

[5] W. Chen, Y. Wang, and S. Yang, *Efficient Influence Maximization in Social Networks*, KDD Conf., (2009).

[6] R. Diestel, Graph Theory (3rd ed.), Berlin, New York, *Springer-Verlag*, (2005).

[7] P. Domingos, and M. Richardson, *Mining the network value of customers*, ACM KDD Conference, (2001).

[8] L. Freeman, *Centrality in social networks: Conceptual clarification*, Social Networks, 1, (1979), pp. 215-239.

[9] D. Kempe, J. Kleinberg, and E. Tardos, *Maximizing the Spread of Influence in a Social Network*, ACM KDD Conf., (2003).

[10] J. Kleinberg, *Authoritative sources in a hyperlinked environment*, JACM 46(5), (1999) pp. 604-632.

[11] J. M. Kleinberg, *The flow of on-line information in global networks*, SIGMOD Conf., (2010).

[12] G. Kossinets, J. M. Kleinberg, and D. J. Watts, *The structure of information pathways in a social communication network*, ACM KDD Conf., (2008).

[13] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. S. Glance, *Cost-effective outbreak detection in networks*, ACM KDD Conf., (2007).

[14] J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst, *Cascading Behavior in Large Blog Graphs*, SDM Conf., (2007).

[15] J. Moody, *Peer influence groups: identifying dense clusters in large networks*, Social Networks, 23(4), (2001). pp. 261-283.

[16] M. E. J. Newman, S. Forrest, and J. Balthrop, *Email networks and the spread of computer viruses*, Phys. Rev. E 66, 035101, (2002).

[17] M. E. J. Newman, *The spread of epidemic disease on networks*, Phys. Rev. E 66, 016128, (2002).

[18] X. Song, C.-Y. Lin, B. L. Tseng, and M.-T. Sun, *Modeling and predicting personal information dissemination behavior*, ACM KDD Conf., (2005).

[19] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos, *Epidemic Spreading in Real Networks: An Eigenvalue Viewpoint*, SRDS, (2003), pp. 25-34.

[20] C. Wang, J. C. Knight, and M. C. Elder, *On computer viral infection and the effect of immunization*, ACM Annual Comp. Security App. Conf., (2000).

[21] D. J. Watts, and P. S. Dodds, Influentials, networks, and public opinion formation, *Journal of Consumer Research*, 34(4), (2007), pp. 441–458.

[22] F. Wu, B. Huberman, L. Adamic, and J. Tyler, *Information Flow in Social Groups, Physica A:*, Statistical and Theoretical Physics, (337)1–2, 1, (2004) pp. 327-335.

[23] W. Chen, C. Wang, and Y. Wang, *Scalable influence maximization for prevalent viral marketing in large-scale social networks*, ACM KDD Conf., (2010), pp. 1029-1038.

[24] S. Brin, and L. Page, *The anatomy of a large-scale hypertextual Web search engine*, Comput. Netw. ISDN Syst., (30)1–7, (1998), pp. 107-117.