J Comb Optim (2018) 35:1202–1240
https://doi.org/10.1007/s10878-018-0252-3

Maximizing misinformation restriction within time and budget constraints

Canh V. Pham[3] · My T. Thai[1,2] · Hieu V. Duong[4] · Bao Q. Bui[4] · Huan X. Hoang[3]

Published online: 21 February 2018

Abstract Online social networks have become popular media worldwide. However, they also allow rapid dissemination of misinformation causing negative impacts to users. With a source of misinformation, the longer the misinformation spreads, the greater the number of affected users will be. Therefore, it is necessary to prevent the spread of misinformation in a specific time period. In this paper, we propose maximizing misinformation restriction (MMR) problem with the purpose of finding a set of nodes whose removal from a social network maximizes the influence reduction from the source of misinformation within time and budget constraints. We demonstrate that the MMR problem is NP-hard even in the case where the network is a rooted tree

✉ My T. Thai
thaitramy@tdt.edu.vn; mythai@cise.ufl.edu

Canh V. Pham
maicanhki@gmail.com

Hieu V. Duong
dvhieubg95@gmail.com

Bao Q. Bui
buiquybao.c500@gmail.com

Huan X. Hoang
huanhx@vnu.edu.vn

1 Division of Algorithms and Technologies for Networks Analysis & Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam

2 Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611, USA

3 University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam

4 Faculty of Information Technology and Security, People's Security Academy Hanoi, Hanoi, Vietnam

at a single misinformation node and show that the calculating objective function is #P-hard. We also prove that objective function is monotone and submodular. Based on that, we propose an $1-1/\sqrt{e}$-approximation algorithm. We further design efficient heuristic algorithms, named PR-DAG to show MMR in very large-scale networks.

Keywords Approximation algorithm · Social networks · Misinformation · Information diffusion

1 Introduction

Online social networks (OSNs) have provided an effective platform for interaction and communication for billions of users. Many people have integrated popular online social sites into their daily lives and considered them as the main source of information. For example, the news about the death of Bin Laden was released on Twitter before the US government officially announced it on social media (Sutter 2017). Furthermore, information spreads on OSNs can have substantial impact on the society, especially political trends. For instance, Facebook and Twitter had great impact on the US Presidential Election in 2008 and the Arab Spring in 2010 (Hughes and Palen 2009; Wolfsfeld et al. 2013).

Unfortunately, rapid dissemination of information on OSNs can be used to spread misinformation to a large amount of users. This may lead to huge economic losses as well as negative impacts on the community in real life. For example, false information about assassination of President Obama has indirectly caused a damage of $136.5 billion to the financial market (Domm 2013). In addition, misinformation on social networks had a significant impact on the US Presidential Election in 2016 (Gentzkow 2017). In order for the OSN to serve its users as a reliable channel of information, it is critical to have effective measures to detect the sources of misinformation and restrict their spread.

Identifying sources of misinformation is a basis for preventing misinformation. Some initial studies have used data mining methods to detect sources of misinformation/rumors (Qazvinian et al. 2011; Kwon et al. 2013). Recently, in order to prevent the outbreak of misinformation and rumors, a commonly used strategy is to block accounts and links which play a vital role in the propagation process. For example, in 2015, Twitter removed 125,000 accounts which was suspected to relate to terrorisms (Yadron 2017). Facebook deleted 30,000 fake accounts reporting rumors in France before the Presidential Election in 2017 (Kottasov 2017).

However, in reality, we cannot remove too many nodes and links because it can affect the spread of information and freedom of speech. Therefore, it is necessary to have an optimal solution for selecting edges and nodes needed to be removed. Information diffusion models provide the basis for the study of solutions to prevent the spread of misinformation in which two most used ones are linear threshold model (LT) and the independent cascade model (IC) (Kempe et al. 2003). Kimura et al. proposed heuristic methods to remove sets of edges from network in order to minimize influence from sources of misinformation (Kimura et al. 2008, 2009). In

LT model, Khalil et al. (2014) proposed an algorithm with an approximation ratio of $1-1/e-\epsilon$ for this problem. Recently, Zhang et al. has designed a strategy to vaccinate sets of edges or nodes in order to reduce the influence of sources of epidemics (Zhang and Prakash 2015, 2014). After that, (Zhang et al. 2016a, c) proposed a strategy of placing monitor machine to prevent the impact of sources of misinformation to important users. In essence, vaccinating and placing monitor on nodes are equivalent to removing nodes from the network, from the abstraction point of view.

A shortcoming of the above studies is that they do not take into account the time constraints when preventing spread of known sources of misinformation. In reality, information is spread from this user to another through only a few hops. The earlier the preventing of spread of misinformation, the smaller the amount of affected users. In addition, cost of removing one user from the network by placing monitor machine or vaccinating nodes are often different. Therefore, in this paper, we investigate a problem, *Maximizing Misinformation Restriction* which seeks the set of nodes to be removed satisfying two conditions: (1) the total cost to remove nodes does not exeed given budget; (2) the number of propagation hops is limited, such that the influence reduction of misinformation source is maximal. Our main contributions are summarized as follows:

- We expand the LT model by combining the constraint of number of propagation step hops, called T-LT model, given a set of source of misinformation nodes, time constraint and limited budget. In this model, we formulate *Maximizing Misinformation Restriction* (MMR) problem that seeks a set of nodes to remove from the network within limited budget and time constraints such that influence reduction of misinformation sources is maximized.
- For complexity, we show that MMR is NP-hard and even if when the network is a rooted tree at single misinformation source node, and prove that calculation of the objective function is #P-hard.
- When the network has a form of tree rooted from only one misinformation source, we designed a *Fully Polynomial - Time Approximation Scheme* (FPTAS) algorithm which is the best approximation algorithm for NP-hard problem. In general case, we prove that the objective function is *monotone* and *submodular*, and thus a Greedy algorithm will have an approximation ratio of $1-1/\sqrt{e}$. We further propose Speed-up Greedy (SG) algorithm by using the state-of-the-art technique in Zhang et al. (2016b). SG provide an approximation ratio of $1-1/\sqrt{e}-\epsilon$. In order to make this solution scalable to billion of nodes and edges, we introduce an efficient heuristic algorithm, called PR-DAG.
- Experiments are performed on real-world social traces of Gnutella, Oregon, Epinions and EU Email datasets showed the performance of our purposed algorithms. In each of these networks, we observe that PR-DAG and SG outperformed baseline methods and they gave the similar result in terms of maximizing the target function while PR-DAG run faster than SG (upto 45 times faster). Experiments also sho that PR-DAG is scalable with EU Email networks, which is large-scale social network contain about 265K nodes and 420K edges.

1.1 Related work

Studying information propagation model is critical for research about influence maximization problem (Kempe et al. 2003) and reducing spread of misinformation/rumors on social networks (Zhang and Prakash 2015; Khalil et al. 2014; Kimura et al. 2008, 2009; Zhang and Prakash 2014; Zhang et al. 2013, 2016a, b; He et al. 2011; Budak et al. 2011; Nguyen et al. 2013). In Domingos and Richardson (2001) studied propagation of information on social networks based on data mining technique. Based on this study, Kempe et al. (2003) proposed two information propagation models which are the Linear Threshold (LT) and the Independent Cascade (IC). In these models, they proposed the problem of Influence Maximization (IM) and a greedy algorithm with a ratio of $1-1/e$. Later, numerous works about proposed scalability and efficiency algorithms have been done (Goyal et al. 2012; Chen et al. 2010a, b; Nguyen and Zheng 2013). In addition, many related diffusion information and variable problems were proposed and studied (Liu et al. 2012; Chen et al. 2012; Zhang et al. 2013; Bhagat et al. 2012) and some studies have extended these two models (Zhang et al. 2013; Liu et al. 2012; Chen et al. 2012).

An important task of preventing misinformation is to identify the source of misinformation or rumors. They can be identified through collecting contents of posts, comments, shares (Qazvinian et al. 2011). Rumors can be identified by using features such as time, structures and language of users (Kwon et al. 2013). In order to prevent the spread of misinformation or bad information, some work proposed a campaign of injecting good information into some nodes in order to fight the misinformation (Budak et al. 2011; Nguyen et al. 2013; He et al. 2011). Budak et al. Budak et al. (2011) proposed a problem of selecting a seed set to create positive information so that the number of users believing in negative information is minimized. He et al. (2011) studied Influence Blocking Maximization problem which chose k nodes to initiate positive information in order to minimize influence of negative information. Nguyen et al. (2013) studied β_T^I-Node Protectors problem with the purpose of seeking the smallest set of nodes with good information to decontaminate misinformation. However, in order to apply all above strategies, we need to know the content of the negative information and create positive information equivalently. Furthermore, it is difficult to persuade users when they already believe in the false information.

Recently, another branch of work used vaccinating/monitoring (Zhang and Prakash 2014, 2015; Zhang et al. 2016a, b) on a set of edges or nodes which are equivalent to solution of removing edges or nodes in the network to reduce the influence of misinformation sources (Khalil et al. 2014). Zhang et al. (2016a) proposed τ-MP problem that prevent propagation of misinformation from the sources to center set nodes at most guaranteed threshold $\tau \in [0, 1]$ by placing monitors on some nodes. They showed that the problem was #P-hard under IC model and proposed greedy algorithm based on cut sets method. However, it is difficult to apply it for a large number of central nodes because the algorithm using a Monte–Carlo sampling technique which takes a long running time. Given a set of infected notes in a network, Zhang and Prakash (2015, 2014) proposed a vaccination strategy for the k remaining nodes so that the number of infected nodes after propagation is minimal under IC model. The similar method have also been applied for groups of

edges and nodes to control propagation of epidemics under LT model (Zhang et al. 2016b).

Most of existing works ignored two important aspects which are the time constraints and the cost of monitoring/vaccinating/removing a set of nodes. Preventing the spread of misinformation within time constraints is essential to limit their harmful effects. Moreover, the recent studies showed that information is spread after few hops from the source and almost negligible in the next steps (Cha et al. 2009; Leskovec et al. 2007). In addition, we considered cost of monitoring/vaccinating/removing each node on network and the total cost is limited.

1.2 Organization

The rest of the paper is structured as follows. We formulate the propagation models and problem definition in Sect. 2. Section 3 introduces some hardness and complexity results. Sections 4, 5 present our proposed algorithms. The experiments on several datasets are presented in Sect. 6. Finally, we give some implication for future work and conclusion in Sect. 7.

2 Model and problem definitions

In this section, we define the diffusion model called Time constraint Linear Threshold (T-LT) based on the traditional LT model. We next formulate the MMR problem, which aims to find a set of nodes to remove from the network with limited budget and time constraints. In Table 1, the frequently used notations are summarized.

2.1 Diffusion model

The most well known models are the LT model and the IC model (Kempe et al. 2003). However, they are not suitable for the time constraints of propagation process. In this subsection, we first describe the standard LT model and then present our extension that incorporates time constraints of the diffusion.

2.1.1 Linear threshold model

Let $G = (V, E, w)$ be a social network with a set of nodes V and a directed edge E, with $|V| = n$ and $|E| = m$. In this paper, the concept of vertex is equivalent to node. Each directed edge $(u, v) \in E$ is associated with an influence weight $w(u, v) \in [0, 1]$ such that $\sum_{u \in N_-(v)} w(u, v) \leq 1$.

Given a subset $S = \{s_1, s_2, \ldots, s_p\} \subset V$ represents a source of misinformation (as the *seed set* in IM problem). In this paper, each node has two states, *active* and *inactive*, $\sigma(S)$ denotes the expected number of activated nodes. In the LT model, the influence cascades in G are illustrated as follows. First, every node $v \in V$ uniformly chooses a threshold $\theta_v \in [0, 1]$, which represents the weighted fraction of u's neighbors that must be active to activate u. Next, the influence propagation happens in round $t = 1, 2, 3...$

Table 1 Table of symbols

Notional	Description
n, m	The number of nodes and the number of edges
$N_-(v), N_+(v)$	The sets of incoming, and outgoing neighbor nodes of v
$d_-(v), d_+(v)$	In-degree, out-degree of v
S	Set of misinformation source nodes, for short is source
$G_d = (V_d, E_d)$	The graph of V_d, E_d which are the set of nodes and edges within d distance from any node in S
$N_E(A)$	The set of incoming and outgoing neighbor edges of A
$\mathcal{G}(G)$	The set of sample graph generating from G
$\mathcal{G}(G \setminus X)$	The set of sample graph generating from $G \setminus X$, $X \subset E$
$\sigma_d(S)$	Influence of S after d hops
$\sigma_d(S, A)$	Influence of S when remove set of nodes $A \in V$ after d hop
$\mathsf{P}(G, u)$	The set of all simple paths starting node u in G
$\mathsf{P}(G, u, v)$	The set of all simple paths starting from u and ending at v
$\mathsf{P}_i(G, u, v), \mathsf{P}_i(G, u)$	The set of paths each of which has the length i in $\mathsf{P}(G, u, v)$ and $\mathsf{P}(G, u)$, respectively
$\sigma_{d,E}(S, X)$	Influence of source S after d hops when removing set of edges $X \in E$
$h(A)$	The *reducetion of influence* of source S when removing set A (objective function)

– At round 1, all nodes in S are activated and the others are inactive.
– At round $t > 1$, an inactive node v is activated if weighted number of its incoming active neighbors is greater than or equal its threshold:

$$\sum_{\text{incoming infected neighbor } u} w(u, v) \geq \theta_v.$$

– Since a node becomes activated, its status remains in the spreading process. The influence propagation stops when no more nodes can be activated.

2.1.2 Time constraint linear threshold model

We are going to describe our extension to the LT model that incorporates time-constraint diffusion processes, called T-LT. Similar to the LT model, in this model, each node chooses a uniform random number in [0, 1] as a threshold, and the spreading process performs in discrete steps. The new aspect in our model is that the influence is limited to the nodes that are within $d \geq 1$ hops from the source. In other words, the difference of T-LT is the propagation process ends after the d propagation steps. The LT model is a special case of T-LT with $d = |V|$. Therefore, T-LT inherits some characteristics of LT model.

Kempe et al. (2003) proved LT to be equivalent to *live-edge* model and *sample graph*. In LT model, for every node $v \in V$, picks at most one random incoming edge, such that the edge (u, v) is selected with probability $w(u, v)$, and no edge is selected

with probability $1-\sum_{u\in N_-(v)} w(u,v)$. The selected edges are called *live* and all other edges are called *blocked*. $\Pr[g|G]$ denotes the probability of sample graph g in G, we have:

$$\Pr[g|G] = \prod_{v\in V} p(v,g,G) \tag{1}$$

where

$$p(v,g,G_d(S)) = \begin{cases} w(u,v), & \text{If } \exists u:(u,v)\in E_g \\ 1-\sum_{u:(u,v)\in E} w(u,v), & \text{Otherwise} \end{cases} \tag{2}$$

$R(g,S)$ denotes the set of nodes reachable from S in g, by claim 2.6 in Kempe et al. (2003), we have:

$$\sigma(S) = \sum_{g\in\mathcal{G}(G)} \Pr[g|G]R(g,S) \tag{3}$$

$R_d(g,S)$ denotes the set of nodes reachable from S in g within distance d. In T-LT model, we notice that v is reachable from source S if and only if: (1) there exits at least one path consisting entirely of live edges (called live-path) from some nodes in S to v, and (2) the collective number of hops along the shortest live-path from S to v is not greater than d. By proof of Theorem 2.11 in Chen et al. (2013), the equivalence between LT and *live-edge* model is with respect to all active sets S_t, where t is the number of step of propagation. Hence, the influence of S after d hops is:

$$\sigma_d(S) = \sum_{g\in\mathcal{G}(G)} \Pr[g|G]R_d(g,S) \tag{4}$$

The influence of S after removal of $A\subset V\setminus S$ is equal to the influence of S on residual graph, the influence of S at this time is:

$$\sigma(S,A) = \sum_{g\in\mathcal{G}(G[V\setminus A])} \Pr[g|G]R_d(g,S) \tag{5}$$

To simplify, instead of considering all sample graphs in G, we only need to consider the sample graphs in $G_d(S)$. We define *benefit* of A is expected reduction of influence of S after removing A, that is,

$$h_d(A) = \sigma_d(S,\emptyset) - \sigma_d(S,A) \tag{6}$$

For convenience, we simplify the symbol $h_d(\cdot)$ by $h(\cdot)$ because G and d is constant. Our purpose is to choose a set A with a given budget such that $h(A)$ is maximal. The definition of the problem is presented in the next subsection.

2.2 Problem definition

Suppose that each node $u \in V$ has a cost $c(u) \geq 0$ for removing from $V \setminus S$ and a limited cost is $L > 0$. When $d = 1$ finding the solution is trivial and does not make much sense, so we only consider the case $d \geq 2$. In this paper, we consider *Maximum Misinformation Restriction* (MMR) which is defined as follows:

Definition 1 (MMR) Given a graph $G = (V, E, w)$ on T-LT model with $d \geq 2$ be a number of hops of propagation, potential misinformation sources $S = \{s_1, s_2, \ldots, s_p\} \subset V$, and a budget $L > 0$. Find the set of vertices A with total cost $c(A) = \sum_{u \in A} c(u) \leq L$ to maximize $h(A)$.

MMR is generalized by the following cases:

- When $d = n$: Maximum Misinformation Restriction on whole process propagation of misinformation source S.
- When $c(u) = 1\ \forall u \in U \subset V, c(u) = +\infty, u \in V \setminus U$, find the set A in the allowed area U to maximize $h(A)$.

3 Hardness and complexity

In this section, we show that MMR is NP-hard even when the graph is the rooted tree by reducing from the *0-1 Knapsack* problem. Furthermore, we prove that the exact calculation of $h(A)$ is #P-hard.

Theorem 1 MMR *is NP-hard in T-LT model, even for* $d = 2$ *and the input graph is the rooted tree.*

Proof To prove that MMR is NP-hard, we will reduce the decision of 0–1 Knapsack problem.
Knapsack Given $2n + 2$ positively integers: $W, K, s_1, \ldots, s_n, c_1, \ldots, c_n$, determine whether there is a sequence $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n$ such that $\sum_{i=1}^{n} w_i x_i \leq W$ and $\sum_{i=1}^{n} c_i x_i \geq K$.
Construction Let $\mathcal{I}_1 = (W, K, w_1, \ldots, w_n, c_1, \ldots, c_n)$ is an instance of Knapsack. We construct a rooted tree $\mathcal{I}_2 = (G, S, d, L)$ which is an instance of MMR as follows (Fig. 1):

- We build only one source node $S = \{s\}$. For each c_i, we construct a direct simple path, in which: the starting node is s, c_i next nodes: $u_{i,1}, u_{i,2}, \ldots, u_{i,c_i}$, the weight of all edges in this path is 1.
- The cost of nodes: $c(u_{1,i}) = 1, c(u_{2,i}) = c(u_{3,i}) = \ldots = c(u_{c_i,i}) = +\infty$, $i = 1 \ldots n$.
- Finally, set $L = W$ and $Z = K$ and $d = \max_{i=1\ldots n}\{c_i\}$.

By the construction, we obtain graph $G = (V, E, w)$ satisfies LT model. Now, we prove that if $\mathcal{I}_1$ has a solution $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ the $\mathcal{I}_2$ has a solution $A = \{u_{i,1} | x_i = 1\}$ such that $h(A) \geq Z$ and vice versa.

($\rightarrow$) Suppose $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is solution of instance $\mathcal{I}_1$. Now with our construction we see that, we chose set $A = \{u_{i,1} | x_i = 1\}$. We have $c(A) = \sum_{i|x_i=1} w_i \leq$

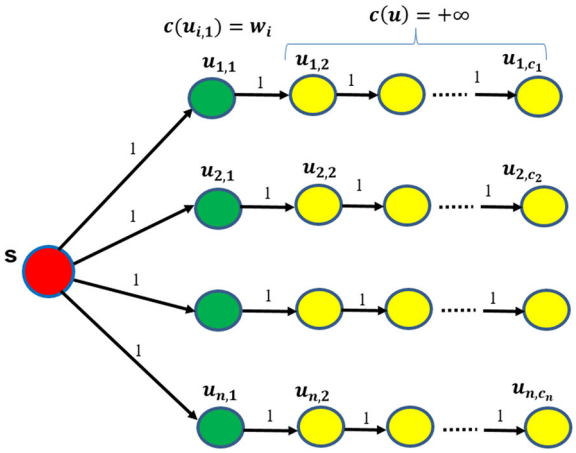

Fig. 1 Reduce from 0 to 1 Knapsack to MMR

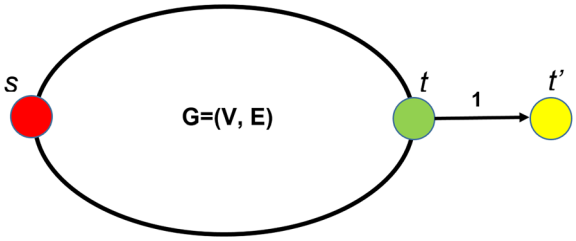

Fig. 2 The constructing instance in the reduction from instance $G = (V, E)$ of $s-t$ paths problem to instance G' of the exact computation of $h(A)$ by adding a new node t'

$W = L$. According LT model, for each path, we select $u_{i,1}$ in set A, the influence from s to another nodes is blocked, implies the reduction of influence is c_i. Therefore, $h(A) = \sum_{i|x_i=1} c_i \geq K = Z$.

($\leftarrow$) Otherwise, if $\mathcal{I}_2$ has the solution A, A can not contain $u_{i,j}$ for $j \geq 2$ since the cost of this node is $+\infty$. Suppose $A = u_{i,1}$, $i \in I$, now we chose a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ for $x_i = 1$ if $u_{1,i} \in A$, otherwise $x_i = 0$. We have $c(A) = \sum_{i|u_{i,1}\in A} w_i = \sum_{i=1}^{n} x_i w_i \leq L = W$ and $h(A) = \sum_{i|u_{i,1}\in A} c_i = \sum_{i=1}^{n} x_i c_i \geq Z = K$ implies $\mathbf{x}$ is solution of $\mathcal{I}_1$ □

Theorem 2 *It is #P-hard to compute $h(A)$ in T-LT model.*

Proof We will reduce from the s-t paths problem defined in the following. Given a directed graph $G = (V, E)$, $|V| = n$, $|E| = m$, computing the number of (directed) paths from node s to node t that visit every node at most once has been proved to be #P-complete (Valiant 1979). From G, we construct a graph G' by adding a new node t' and adding an edge from t to t' with weight $w(t, t') = 1$. The number of vertices of G is $n + 1$. Let Δ be the maximum in-degree of any node in G'. For $e \neq (t, t')$, $e \in E$ we set $w(e) = w = 1/\Delta$, this assumption satisfies the LT model since the total of in-neighbour weight is less than 1. We set $S = \{s\}$, $A = \{t'\}$ and $d = n + 1$. We will show that if for any eligible w computing $h(t')$ on G' is solvable, the s-t problem is also solvable (Fig. 2).

By the equivalence given claim 2.6 in Kempe et al. (2003), we have:

$$\sigma_d(S, \emptyset) = \sum_{x \in \mathsf{P}(G',s)} \prod_{e \in x} w(e). \tag{7}$$

$$\sigma_d(S, A) = \sum_{x \in \mathsf{P}(G' \setminus \{t'\},s)} \prod_{e \in x} w(e). \tag{8}$$

Eliminate the same elements in the two above equations, the remaining paths contain node t', we have

$$h(t') = \sigma_d(S, \emptyset) - \sigma_d(S, A) = \sum_{x \in \mathsf{P}(G',s,t')} \prod_{e \in x} w(e) \tag{9}$$

Since each simple path from s to t' in G' contains a simple path from s to t in G, we have:

$$h(t') = \sum_{x \in \mathsf{P}(G',s,t')} \prod_{e \in x} w(e) = \sum_{i=0}^{n-1} \sum_{x \in \mathsf{P}_i(G,s,t)} \left(w(t,t') \prod_{e \in x} w(e) \right) = \sum_{i=0}^{n-1} w^i \cdot \alpha_i \tag{10}$$

where $\alpha_i = |\mathsf{P}_i(G, s, t)|$. For Eq. (10), we can set weight w to n distinction values $w_1, w_2, \ldots, w_n$ and we have value of $h(t')$ corresponding to w_i. Hence, we obtain a set of n linear equations with $\{\alpha_0, \alpha_2, \ldots, \alpha_{n-1}\}$ as variables. The matrix of this equation is $M_{n \times n} = \{m_{ij}\}$ and $m_{ij} = w^i, i, j = 0, .., n-1$ so this is Vandermonde matrix and we can easily to compute the unique solution $\{\alpha_0, \alpha_2, \ldots, \alpha_{n-1}\}$ for the linear system of equations. The total of s-t paths in G is $\sum_{i=0}^{n-1} \alpha_i$. If $\mathcal{A}$ is a polynomial-time algorithm which can calculate exactly $h(t')$, we can use $h(t')$ to calculate s-t paths, implying that the exact calculaiton of $h(t')$ is at least as hard as s-t paths problem □

4 Approximation algorithms

In this section, we introduce some approximation algorithms for MMR problem. First, we provide a Fully Polynomial Time Approximation Scheme (FPTAS) algorithm for the case when the network is a rooted tree at single misinformation source node. In the general case, we designed a fast greedy algorithm with a ratio of $1 - 1/\sqrt{e}$ as the objective function is proved to be *monotone* and *submodular*.

4.1 FPTAS for rooted tree

In this section, we consider *tree version* of MMR (called T-MMR) that means the network is a tree rooted at a source node I. We designed FPTAS for T-MMR, the algorithm has the best approximation ratio $(1 - \epsilon)$ for NP-hard problem.

4.1.1 Calculate benefit of nodes

When the graph is a tree rooted at I, we consider the sub-tree rooted I that each node has a maximum depth d, called T_I. If we select node u, the influence of I will not reach any descendant nodes of u. $Inf(u, v)$ denotes the influence from u to v when u is activated, $D(u)$ is the set of descendant nodes of u on T_I, we have:

$$h(v) = Inf(I, v) + Inf(I, v) \cdot \sum_{u \in D(v)} Inf(v, u) \tag{11}$$

To calculate $h(.)$ we propose a recursive algorithm based on Depth-First Search (DFS) described in Algorithm 1.

Algorithm 1: Calculate benefit of node u for tree CalBen(T_I, u)

```
Data: A tree T_I = (V_I, E_I, w), I.
Result: benefit h(u), u ∈ V
1  h ← 1
2  if u is not a leaf then
3  |   for v is child of u do
4  |   |   h ← h + CalBen(v) · w(u, v)
5  |   end
6  end
7  h ← h · Inf(I, par(u)) · w(par(u), u)
8  Return h
```

Lemma 1 *Algorithm 1 has a complexity is $\mathcal{O}(n_T)$, where n_T is number of nodes in T_I.*

Proof The steps of the algorithm is similar to the algorithm DFS so it has a complexity $\mathcal{O}(m_T + n_T) = \mathcal{O}(n_T)$ (due to T_I is a tree) □

4.1.2 FPTAS for T-MMR

Based on the tree structure of the problem, we used *dynamic programming method* to design a *Fully Polynomial Time Approximation Scheme* (FPTAS). This algorithm is divided into two phases. In the first phase, we find sub-tree rooted at I with the depth d to satisfy the time constraint of the problem. Next, we standardize benefit value of nodes as well as the flow values. Basically, we try to bound them by a polynomial in n and $\frac{1}{\epsilon}$. In the second phase, we use the dynamic programming to find the solution in polynomial time. The algorithm is summarized in Algorithm 2 (Fig. 3).

Phase 1: Preprocessing We find sub-tree root at I has the depth at most d. Next we will calculate the benefit of nodes by Algorithm 1 and standardize the benefit as shown in line 4 in Algorithm 2 then $h(u)$ is scaled down by a factor K. This preprocessing step ensures that all $h(u)$ is integer between 0 and $\lceil \frac{n}{\epsilon} \rceil$.

Phase 2: Dynamic Programming In the second phase, we use dynamic programming to find an optimal solution for the MMR problem instance (G, I, h) Now, given the

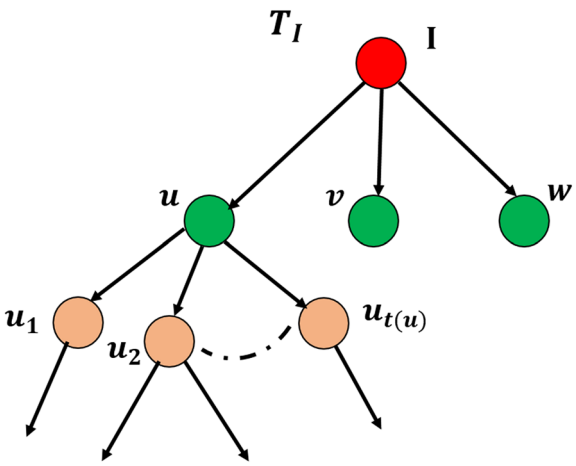

Fig. 3 The tree T_I root at I

tree $G = (V, E)$ with $|V| = n$, and the root I. In order to describe the dynamic algorithm, we use the following the notations:

- T^u: The sub-tree rooted at u in T_I, with the set of vertices V^u and the set of edges E^u.
- $t(u)$: The outcoming degree of node u.
- $u_1, u_2, \ldots, u_{t(u)}$: represent children of u.
- T_i^u: The i-th sub-tree of tree rooted at u i.e, the subtree rooted at i-th u's child.

We define the following recursion functions:

- $F^u(p)$: The minimum total costs when removing a set of nodes in T^u so that the total benefits gained on T^u is at least p.
- $F_i^u(p)$: The minimum total costs to remove a set of nodes in T_i^u so that the total benefits gained on T_i^u is at least p.
- $H_i^u(p)$: The minimum total costs when removing the set of nodes in subtrees $\{T_1^u, T_2^u, \ldots, T_i^u\}$ so that the total benefits gained is at lest p, where $i = 1 \ldots t(u)$.

The core of the dynamic algorithm is to compute $F^u(p)$ and $H_i^u(p)$ through the following recursions:

$$F^u(p) = \begin{cases} \min\{H_{t(u)}^u(p), c(u)\}, & \text{If } h(u) \geq p \\ H_{t(u)}^u(p), & \text{Otherwise} \end{cases}, \forall p = 1 \ldots \lceil \frac{n^2}{\epsilon} \rceil \tag{12}$$

$$F_i^u(p) = \min_{q=1\ldots p} \{H_{i-1}^u(q) + F_i^u(p-q)\}, \forall p = 1 \ldots \lceil \frac{n}{\epsilon} \rceil, i = 1 \ldots t(u). \tag{13}$$

The basic cases are as follows.

$$F^u(p) = \begin{cases} 0, & p \leq 0 \\ +\infty, & p > \lceil \frac{n^2}{\epsilon} \rceil \end{cases} \tag{14}$$

$$F_i^u(p) = \begin{cases} 0, & p \leq 0 \\ +\infty, & p > \lceil \frac{n^2}{\epsilon} \rceil \end{cases} \tag{15}$$

If node u is a leaf.

$$F^u(p) = \begin{cases} c(u), & \text{If } h(u) \geq p \\ +\infty, & \text{Otherwise} \end{cases} \tag{16}$$

Finally, the maximum objective for instance (G, I, h') is given at the root u by:

$$OPT(T, I, h') = \max\{p | F^u(p) \leq C\} \tag{17}$$

Lemma 2 *Algorithm 2 finds an optimal solution for* T-MMR *instance* (G, I, h') *has the complexity of* $\mathcal{O}(\epsilon^{-2}n^5)$

Algorithm 2: FPTAS for T-MMR

Data: $G = (V, E, w), I, d, \epsilon > 0$.
Result: A
1 *Phase 1: Preprocessing*
2 Find sub-tree T_I of G root I has depth d.
3 CalBen$(T_I, u), \forall u \in T_I$
4 $B = \max\{h(v) | v \in V, c(v) \leq L\}$, $K = \frac{\epsilon B}{n}$
5 Let $h'(u) = \lfloor \frac{h(u)}{K} \rfloor$
6 *Phase 2: Dynamic Programming algorithm*
7 Compute $F^u(p)$, $F_i^u(p)$ using the recursions (12) and (13)
8 Find an optimal solution, call A', by tracing from $\max\{p | F^u(p) \leq L\}$
9 Return A'

Proof The accuracy of the dynamic programming algorithm comes from the sub-optimal structure of the problem. For calculation $F^u(p)$, the major portion of running time is to compute $F_i^u(p)$. Since node u has maximum $n-1$ children and $q \leq p \leq \lceil \frac{n^2}{\epsilon} \rceil$. The running time to compute is $\mathcal{O}(n \cdot \lceil \frac{n^2}{\epsilon} \rceil \cdot \lceil \frac{n^2}{\epsilon} \rceil) = \mathcal{O}(\epsilon^{-2}n^5)$ □

Theorem 3 *Algorithm 2 is FPTAS for* T-MMR.

Proof Let A^* be an optimal solution of T-MMR instance $\mathcal{I}_1 = (G, I, h)$ with the objective value $OPT = h(A^*)$. A' is an optimal solution of T-MMR instance $\mathcal{I}_2 = (G, I, h')$ by running Algorithm 2 with the objective value $OPT' = h'(A)$, with $h'(.)$ is value of for $\mathcal{I}_2$. We see that A' is feasible solution of $\mathcal{I}_1$ since it satisfies the condition that the highest cost is L. We need to prove:

$$h(A') \geq (1 - \epsilon) \cdot h(A^*) \tag{18}$$

Due to rounding down $h'(u)$ in line 4 in Agorithm 2, we have:

$$h(u) \geq K \cdot h'(u), h'(u) \geq \frac{h(u)}{K} - 1$$

Therefore:

$$h(A') \geq K \cdot h'(A') \geq K \cdot h'(A^*) \geq h(A^*) - n \cdot K = OPT - \epsilon \cdot M$$

Since we filtered nodes u with $c(u) > L$, we have $OPT \geq M$. Therefore,

$$h(A') \geq OPT - \epsilon \cdot M \geq (1 - \epsilon) \cdot OPT$$

Thus the objective of A' is within a factor $1 - \epsilon$ of OPT i.e. Algorithm 2 is $(1 - \epsilon)$ approximation for $0 < \epsilon < 1$ for T-MMR. Combining lemma 2 this alg. has a time complexity of $\mathcal{O}(\epsilon^{-2} n^5)$ so it is a FPTAS □

4.2 Approximation algorithm for general case

We next introduce our $(1 - 1/\sqrt{e})$-approximation algorithm. We first present the following lemmas and theorems.

Lemma 3 *$\sigma_{d,E}(S, X)$ is monotonically decreasing function of the set of edges X to be deleted.*

Lemma 4 *The function $\sigma_{d,E}(S, X)$ is supermodular, i.e. for $X \subseteq Y \subset E$ and $\forall e \in E \setminus Y$ then:*

$$\sigma_{d,E}(S, X) - \sigma_{d,E}(S, X \cup \{e\}) \geq \sigma_{d,E}(S, Y) - \sigma_{d,E}(S, Y \cup \{e\}) \tag{19}$$

The proofs of Lemmas 3 and 4 are presented in "Appendix".

Theorem 4 *The benefit function $h(A)$ is monotone:*

$$h(A) \leq h(T) \tag{20}$$

and submodular function:

$$h(A \cup \{v\}) - h(A) \geq h(T \cup \{v\}) - h(T), \tag{21}$$

where $\forall A \subseteq T \subset V, v \notin T$

Proof Removing a vertex u is equivalent to the influence passing through edges adjacent to it blocked. Hence, $\sigma_d(S, A) = \sigma_{d,E}(N_E(A))$. For vertex $v \in V$ then $N_E(A) \subseteq N_E(A + \{v\})$ combining with Lemma 3, we obtain that $\sigma_d(S, A)$ is monotonically decreasing which infers that $h(A)$ is monotonically increasing.
Denote $E_{T,v} = N_E(T + \{v\}) \setminus N_E(T)$, $E_{A,v} = N_E(A + \{v\}) \setminus N_E(A)$. Due to $A \subseteq T$ the set of edges adjacent to v but not adjacent to any vertices in T is subset of set of edges adjacent with v but not adjacent any vertices in A, we have $E_{T,v} \subseteq E_{A,v}$ and $N_E(A) \cup E_{T,v} \subseteq N_E(A) \cup N_E(v) = N_E(A + \{v\})$. Using Lemmas 4 and 3, we have:

$$\begin{aligned}\sigma_d(S, A) - \sigma_d(S, A + \{v\}) &= \sigma_{d,E}(S, N_E(A)) - \sigma_{d,E}(N_E(A + \{v\})) \\ &\geq \sigma_{d,E}(S, N_E(A)) - \sigma_{d,E}(S, N_E(A) \cup E_{T,v}) \\ &\geq \sigma_{d,E}(S, N_E(T)) - \sigma_{d,E}(S, N_E(T) \cup E_{T,v}) \\ &= \sigma_d(T) - \sigma_d(T + \{v\})\end{aligned}$$

Combine with the definition of (Eq. 5) we have:

$$h(A + \{v\}) - h(A) \geq h(T + \{v\}) - h(T)$$

This has completed the proof □

In Algorithm 3, we make a greedy strategy. In each step, we select a node u that maximizes the *marginal benefit* over the cost ratio if the cost of u is not greater than the remaining budget. Let A be the set of currently selected vertices, the marginal benefit of node u over the cost ratio is calculated by the formula:

$$\delta(v) = \frac{h(A \cup v) - h(A)}{c(v)}, \forall v \in V \setminus A \tag{22}$$

The algorithm terminates when at least one of the two conditions occurs: no budget remains, or no node can be added to A. In the case that all nodes have the

Algorithm 3: Greedy algorithm

Data: $G = (V, E, w), L, d, S$.
Result: A
$A \leftarrow \emptyset$;
$U \leftarrow N_d(S)$;
repeat
  $\delta(v) = \frac{h(A \cup v) - h(A)}{c(v)}$
  $u = \arg\max_{v \in U, c(v) \leq L} \delta(v)$
  if $c(A \cup u) \leq L$ **then**
    $A \leftarrow A \cup u$
  end
  $U \leftarrow U \setminus u$
until $U = \emptyset$;
Return A;

same cost, Algorithm 3 achieves an approximation ratio of $1 - 1/e$. However, with different node costs, Algorithm 3 can have unbounded approximation ratio and it gives even bad results. For example, considering the directed network containing $t + 2$ nodes $V = \{I, u, v_1, v_2, \ldots, v_t\}$, I is a source node, the set of edges $E = \{(I, u), (I, v_1), (v_1, v_2), (v_2, v_3), \ldots, (v_{t-1}, v_t)\}$ and the weight of all edges is equal to 1. Let the cost $c(u) = 1 - \epsilon, c(v_i) = t, \forall\, i = 1 \ldots t$ and the budget $L = t$. The optimal solution is node v_1 which achieves a benefit of t. Algorithm 3 only chooses node u since it has the maximum benefit gained over cost ratio $1/(1 - \epsilon)$, resulting

benefit is 1. Next, we inherit idea of Khuller et al. (1999) to modify the Algorithm 3 to achieve a constant approximation ratio. First, we remove all nodes that cost more than the budget because they are not feasible solutions. Let A_1 be selected set of nodes by Algorithm 3. We will consider another candidate solution v_{max}, which is the node that has the largest benefit. We compare the benefit of A_1 and v_{max}, then output the one with higher benefit. The process is summarized in Algorithm 4.

Theorem 5 *Algorithm 4 provides a* $(1-\frac{1}{\sqrt{e}})$*-approximation to find max* $h(S)$ *problem.*

Algorithm 4: Improved Greedy Algortihm (IGA)

Data: $G = (V, E, w), L, d, S$.
Result: A
$U \leftarrow$ remove all nodes having cost greater than L from V
$A_1 =$ Result of Greedy;
$v_{max} = \arg\max_{u \in U} h(v)$
$A = \arg\max\{h(A_1), h(v_{max})\}$
Return A;

Proof Straightforward based on Khuller et al. (1999). □

Complexity Let R be the maximum time needed to calculate the value of $h(A), \forall A \subseteq V$. Algorithm 3 runs in $\mathcal{O}(n^2R)$ where n is the number of nodes, k is the number of iterators. u_{max} can be determined in $\mathcal{O}(nT)$ time. Hence, Algorithm 4 runs in $\mathcal{O}(n^2R)$. Note that we can improve the approximation ratio to $1-1/e$ by inheriting greedy with partial heuristic enumeration (Khuller et al. 1999). However, the time complexity of $\mathcal{O}(n^4R)$ will become unfeasible. The difficulty in implementing the Algorithm 4 is computing. Kempe et al. (2003) used Monte–Carlo simulations method to estimate the influence function and this method can be applied for the algorithm. However, it is hard to apply to large graphs due to its high complexity. Therefore, to speed up the Algorithm 4, we will propose Speed-up Greedy (SA) algorithm based on two aspects: (1) simplifying the source nodes S into super source node I, and (2) and fast updating the function based on characteristics of generated samples.

4.2.1 Speed-up of the improved greedy algorithm

In this subsection, we apply GREEDY-LT method proposed by Zhang et al. (2016b) which is the state-of-the-art method to Speed-up IG algorithm. We first simplify the MMR problem by merging set source $S = \{s_1, s_2, \ldots, s_p\}$ into a *supper source node* I. In T-LT model if a healthy node has multiple neighbors which are source nodes, it will have a new edge weight which would be the sum of weights. For example, if node u has two source neighbors s_1 and s_2, the weight of merged node I would be: $w(I, u) = w(s_1, u) + w(s_2, u)$. The details of this method are presented in Algorithm 5 and equivalence between the before and after instances of MMR is proved in Proposition 1.

Proposition 1 *Given an instance of the* MMR *instance* (G, S, w) *under T-LT model, Algorithm 5 outputs an equivalent instance* (G', I, w') *where* I *is the only source of misinformation node in the new graph* G'.

Proof We will prove this proposition by induction. If a node v connects to two source nodes u_1 and u_2. In T-LT model, the influence of u_1 and u_2 to v will be $w(u_1, v) + w(u_2, v)$ which is the same as the total influence of output of Algorithm 5 (line 9). When v has $i + 1$ incoming neighbours source nodes $u_1, u_2, \ldots, u_{i+1}$. Denote by $w'_i(I, v)$ is influence after merging i nodes. When merging a new source node u_{i+1}, the total influence is $w'_{i+1}(I, v) = w'_i(I, v) + w(u_{i+1}, v)$ which is the same as the output of Algorithm 5 (line 9) □

Algorithm 5: Mere source nodes S into supper source I - Merge(G, S)

```
   Data: G = (V, E, w), S.
   Result: G', w' and supper source node I
1  G' ← G
2  Add node I to G'
3  for s ∈ S do
4      if there exits edge (s, v) then
5          if (I, v) ∉ G' then
6              Add edge (I, v) to G'
7              w'(I, v) = w(s, v)
8          else
9              w'(I, v) = w'(I, v) + w(s, v)
10         end
11         Remove (s, v) from G'
12     end
13 end
14 Remove all nodes in S from G'
15 Return G', I
```

Complexity As the same method in Zhang and Prakash (2015), the complexity of alg. 5 is $\mathcal{O}(p + |N(S)|)$, where $|N(S)|$ is number of neighbours of S.

Next, to speed up Algorithm 4, we propose the Speed-up Greedy algorithm (SGA) summarized in Algorithm 6. Firstly, we merge S into a supper source node I and we obtain the new graph is G' (line 2). Let $\mathcal{R}$ be the set of η samples generated from G'. In each sample graph $g \in \mathcal{G}(G')$, all nodes reachable from I is a tree root at I. Since the time constraint is d, denoted $I_I^d \sim g$ is sub-tree of T root at I with height at most d, Let $\mathcal{L}$ denote the set of η sample graphs and $T = \{T_I^d \sim g | g \in \mathcal{G}(G')\}$, $h(u, T_I)$ denotes the benefit of node u in T_I^d, we have $h(u, T_I) = |\{v | v \in subtree(u)\}|$ is the number of nodes that under the sub-tree root at u (including u) in T_I. After that, we apply the framework of Algorithm 4 with the marginal benefit of node u over tree T_I is:

$$h(I, T_I^d) - h(I, T_I^d \setminus u) = h(u, T_I^d) \tag{23}$$

The problem of computing the $h(\cdot)$ exactly has been shown to be #P-Hard. Thus we apply the Sample Average Approximation (SAA) framework to approximate the function $h(\cdot)$ in near-linear time by using empirical average over set $\mathcal{L}$. That is,

$$h(A) \approx \hat{h}(A) = \frac{1}{\eta} \sum_{T_I^d \sim g, g \in \mathcal{L}} h(I, T_I^d \setminus \{A\}) \tag{24}$$

The marginal benefit of node u on G is approximated by the average marginal benefit for all tree $T_I^d \in \mathcal{L}$.

$$\begin{aligned} \delta(A, u) &\approx \hat{\delta}(A, u) = \hat{h}(A) - \hat{h}(A \setminus \{u\}) \\ &= \frac{1}{\eta} \sum_{T_I^d \sim g, g \in \mathcal{L}} h(I, T_I^d \setminus \{A\}) - h(I, T_I^d \setminus \{A \cup u\}) \end{aligned} \tag{25}$$

We choose the node that has the largest marginal benefit $\hat{\delta}(A, u)$ in line 7 and choose the set of nodes A_1 by framework of Algorithm 3 (lines 8–19). We remove the node which is picked into A_1 and update function $h(u, T_I), u \in T_I$ (line 16) in two cases as follows: (1) for children of u, we can remove them because they are not connected to I, (2) for any ancestor v of u, $h(v, T_I^d \setminus u) = h(v, T_I^d) - h(u, T_I^d)$, this task can be done in $\mathcal{O}(1)$. The algorithm has an approximation ratio of $1 - 1/\sqrt{e} - \epsilon$ where ϵ is the approximation factor for estimating $h(\cdot)$.

Complexity of SG *algorithm* Denoted $m_d = |E_d|, n_d = |V_d|$. Generating set tree T can be done by using BFS algorithm with the depth d, this task takes $\mathcal{O}(\eta(m_d + n_d))$ (line 4). Calculating $h(T_I, u), \forall u \in T_I$ can be done using Algorithm 1, the running time is $\mathcal{O}(\eta n_d)$. For greedy phase, in each iterator, choosing node u which has the largest marginal benefit needs $\mathcal{O}(n_d)$, updating the set of trees $\mathcal{L}$ needs $\mathcal{O}(\eta n_d)$. Hence, the total time of this phase is $\mathcal{O}(k_1 \eta n_d)$ where k_1 is the number of iterations of greedy pharse (lines 8–19). Therefore, Algorithm 6 runs in $\mathcal{O}(\eta(m_d + k_1 n_d))$.

5 DAG-based algorithm

In the previous section, we introduced the approximation algorithms. However, difficult to apply these algorithms for large networks due to the fact that it estimate $h(A)$ based on Monte–Carlo simulation method which runs for long time. Therefore, in this section, we will introduce an effective heuristic algorithm which is scalable for large networks. First, we are going to extract *directed acyclic graph* (DAG) from original graph and apply it to estimate the influence propagation. Whereby, we will provide a metric measuring the role of a node in misinformation propagation, called *propagation role*. Next, we put it into framework of Algorithm 4 instead of marginal benefit. To speed-up algorithm, we narrow candidate for solution set in DAG. The details of this algorithm are described in next subsections.

Algorithm 6: Speed-up Greedy (SG) algorithm

Data: Graph $G = (V, E, w)$, source $\mathcal{S}$, budget $L > 0$
Result: set of nodes A

$U \leftarrow V$
$(G', I) \leftarrow \mathsf{Merge}(G, S)$
foreach G' **do**
  Grenerate set η live-edge graphs $\mathcal{L}$ and set tree T.
  For each $T_I^d \in T$, calculate $h(u, T_I^d)$ for all $u \in T_I^d$ (Algorithm 1)
end
$v_{max} \leftarrow \arg\max_{u \in V, c(u) \le L} \frac{1}{\eta} \sum_{T_I \in \mathcal{L}} \left(h(I, T_I^d) - h(I, T_I^d \setminus v) \right)$
repeat
  $c_{min} \leftarrow \arg\min_{v \in V} c(v)$
  If $c_{min} + c(A_1) > L$ then break
  $u \leftarrow \arg\max_{v \in V} \frac{1}{c(v)} \frac{1}{\eta} \sum_{T_I^d \in T} \left(h(I, T_I^d) - h(I, T_I^d \setminus v) \right)$
  $U \leftarrow U \setminus \{u\}$
  if $c(A_1) + c(u) \le L$ **then**
    $A_1 \leftarrow A_1 \cup \{u\}$
    foreach $T_I^d \in T$ **do**
      If $u \in T_I^d$, remove node u and update $h(v, T_I^d), \forall v \in T_I^d$.
    end
  end
until $U = \emptyset$;
$A \leftarrow \arg\max_{v_{max}, A_1} \{h(u_{max}), h(A_1)\}$
return A;

5.1 DAG construction

Recent studies have shown that the influence of a node is very small for a long path (Chen et al. 2010a) so we ignore these paths. On the other hand, the union of maximum influence paths is a good way to approximate the influence (Chen et al. 2010a, 2012, 2010b; Nguyen and Zheng 2013). Furthermore, for the time constraint of d in the MMR problem, we only consider paths has length at most d. Therefore, we use the set of paths which have two characteristics above as a basis for building the DAG. The following definitions are the basis for our method.

Definition 2 (*Influence path*) For a path $P(u, v) = \{u = x_1, x_2, \ldots, x_l = v\}$, of length l from a node u to v, define the influence of the path, $\mathsf{Inf}(P(u, v))$, as:

$$\mathsf{Inf}(P(u, v)) = \prod_{i=1}^{l-1} w(x_i, x_{i+1}) \tag{26}$$

Definition 3 (*Maximum influence path*) The Maximum Influence Path $MIP(G, u, v)$ from u to v is defined as:

$$MIP(G, u, v) = \arg \max_{P \in \mathsf{P}(G,u,v)} \{\mathsf{Inf}(P)\} \tag{27}$$

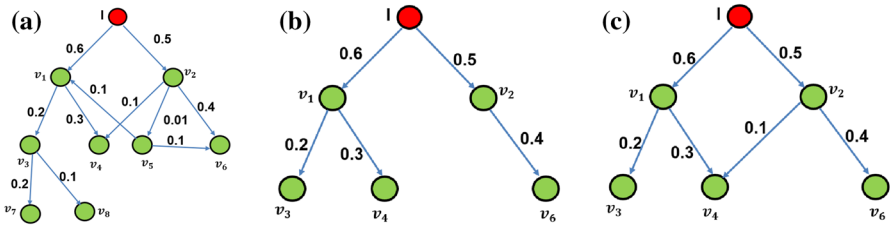

Fig. 4 DAG construction by Algorithm 7: **a** graph G' with source node I, **b** $MIOA(G, I, d, \theta)$ for $d = 2, \theta = 0.051$, **c** DAG was constructed from $MIOA(G, I, d, \theta)$ by add a valid edge (v_2, v_4)

Definition 4 (*Maximum influence path within length constraint*) The Maximum Influence Path $MIP_d(G, u, v)$ from u to v within length d is defined as:

$$MIP_d(G, u, v) = \arg \max_{P \in \mathsf{P}_1(G,u,v) \cup \mathsf{P}_2(G,u,v) \ldots \cup \mathsf{P}_d(G,u,v)} \{\mathsf{Inf}(P)\} \tag{28}$$

$MIP_d(G, u, v)$ is different from $MIP(G, u, v)$. For $MIP_d(G, u, v)$, we consider all paths from u to v which have the maximum length of d and then choose the path that has the greatest influence. Based on $MIP_d(G, u, v)$, we define *Maximum Influence Path Out-Arborescence with Length Constraint* as follows:

Definition 5 For a graph G, an influence threshold θ, the Maximum Influence Out-Arborescence of a node $u \in V$, $MIOA(G, u, d, \theta)$ with length of d, is:

$$MIOA(G, u, d, \theta) = \bigcup_{v \in V, \mathsf{Inf}(MIP_d(G,u,v)) \geq \theta} MIP_d(G, u, v) \tag{29}$$

$MIOA(G, u, \theta)$ is defined as the union of all $MIP_d(G, u, x)$, $\forall x \in V$ with influence path greater than a threshold θ and the maximum length of d. $MIOA(G, u, d, \theta)$ gives the local influence regions (given lower bound) of u within d hops, so with different values of θ we have different local influence regions sizes. $MIOA(G, u, d, \theta)$ can be computed by modifying Dijkstra algorithm in graph G_d with edge weight $-\log w(u, v)$ for edge (u, v).

Now, we use $MIOA(G, I, d, \theta)$ to build the DAG and use it to approximate the influence from node to node in graph. DAG is a finite directed graph with no directed cycles which has at least a topological ordering. In the $MIOA(G, u, d, \theta)$ tree, we add edges from the node with low depth to node with high depth (or high height to low height) in original graph, we obtain a DAG. $d(I, u)$ denotes the depth of node u on the tree $MIOA(G, I, d, \theta)$. The DAG constructing procedure is summarized in Algorithm 7 (Fig. 4).

In DAG, we design a metric to measure the propagation role of a node u in the diffusion from source I. The metric combines the two following aspects: (1) influence from I to u, and (2) influence from u to the other nodes in DAG. We define them as $f_{in}(u)$ and $f_{out}(u)$ respectively,

Algorithm 7: Construct DAG $\mathcal{D}$ from $G = (V, E, w)$ - $\mathsf{DAG}(G, S)$

Data: A graph G, misinformation source S and threshold θ.
Result: DAG $\mathcal{D}$
$(G', I) \leftarrow \mathsf{Merge}(G, S, \theta)$
$\mathcal{D} = MIOA(I, G', d, \theta)$
foreach *edge* $e = (u, v) \in G' | u, v \in \mathcal{D}$ **do**
  if $d(I, u) < d(I, v)$ **then**
    $\mathcal{D} = \mathcal{D} \cup (u, v)$
  end
end
return $\mathcal{D}$

$$f_{in}(u) = \sum_{P \in \mathsf{P}(\mathcal{D}, I, u)} \mathsf{Inf}(P) \tag{30}$$

$$f_{out}(u) = \sum_{v \in U} \sum_{P \in \mathsf{P}(\mathcal{D}, u, v)} \mathsf{Inf}(P) \tag{31}$$

Finally, the *propagation role* of node u defined is,

$$r(u) = f_{in}(u) \cdot f_{out}(u) \tag{32}$$

For example, in Fig. 6, $f_{in}(v_1) = w(I, v_1) = 0.6$, $f_{out}(v_1) = 1{+}0.2{\cdot}1{+}0.3{\cdot}1 = 1.5$, $r(v_1) = 0.6{\cdot}1.5 = 0.9$. Similarly, $r(v_2) = 0.75$, $r(v_3) = 0.12$, $r(v_4) = 0.23$, $r(v_6) = 0.2$. Chen et al. (2010b) design a linear-time to estimate influence from seed set node

Algorithm 8: Calculate the Propagation Role of all nodes in $\mathcal{D}$

Data: DAG $\mathcal{D}$, supper source node I
Result: $r(u), \forall u \in \mathcal{D}$
$f_{in}(u) \leftarrow 0$, $f_{out}(u) \leftarrow 1, \forall u \in \mathcal{D}$
$f_{in}(I) = f_{out}(I) \leftarrow 1$
$topoList \leftarrow \emptyset$
Topologically sort all nodes from I in $\mathcal{D}$ into a sequence $topoList$ based on DFS
foreach $u \in topoList$ *(from the first to the last)* **do**
  foreach $v \in N_{-}(u)$ **do**
    $f_{in}(v) \leftarrow f_{in}(v) + f_{in}(u) \cdot w(u, v)$
  end
end
foreach $u \in topoList$ *(from the last to the first)* **do**
  foreach $v \in N_{+}(u)$ **do**
    $f_{out}(v) \leftarrow f_{out}(v) + f_{out}(u) \cdot w(v, u)$
  end
end
return $f_{in}(u) \cdot f_{out}(u), u \in \mathcal{D}$

on DAG. We use this method to calculate $f_{in}(\cdot)$, $f_{out}(\cdot)$ and $r(\cdot)$. The procedure is showed in Algorithm 8. We first assign $f_{in}(I) = f_{out}(I) = 1$, $f_{in}(u) = 0$, $f_{out}(u) = 1, \forall u \in \mathcal{D} \setminus I$ and topological sort all nodes in $\mathcal{D}$ from I using DFS. We then calculate

the influence from I to each node in order from first to last (lines 5–9). We use node u to update the influence from I to its out-neighbour v (line 7). For $f_{out}(u)$, we do the same above works from the last to the first of list. This work can be performed concurrently with DFS topological sort.

5.2 Algorithm description

The heuristic algorithm is proposed based on a combination of the following aspects: (1) constructing DAG from original graph and calculating the propagation role measure of each node 8, (2) using the framework of Algorithm 4 in which margain benefit be replaced by propagation role. The algorithm is presented in Algorithm 9. The

Algorithm 9: Remove nodes based Propagation Role in DAG (PR-DAG)

Data: Graph $G = (V, E, w)$, $\mathcal{S} = \{s_1, s_2, \ldots, s_q\}$, budget $L > 0, \theta, d$
Result: set of nodes A.

1. Construct graph $\mathcal{D} = \mathsf{DAG}(G, S, \theta)$ (alg. 7)
2. Calculate $r(u), \forall u \in \mathcal{D}$ (alg. 8)
3. $U \leftarrow V_{\mathcal{D}} \setminus I$
4. $v_{max} \leftarrow \arg\max_{u \in U, c(v) \le L}(f_{in}(u) \cdot f_{out}(u))$
5. $G^{cur} \leftarrow G'$
6. **repeat**
7. $\quad c_{min} \leftarrow \arg\min_{v \in U} c(v)$
8. $\quad$ If $c_{min} + c(A_1) > L$ then break;
9. $\quad$ Construct graph $\mathcal{D} = \mathsf{DAG}(G^{cur}, I)$ (alg. 7)
10. $\quad$ Calculate $f_{in}(u)$ and $f_{out}(u), \forall u \in \mathcal{D}$ (alg. 8)
11. $\quad rc_{max} \leftarrow 0; u_{max} \leftarrow null$
12. $\quad$ **foreach** $u \in U$ **do**
13. $\quad\quad$ **if** $c(A_1) + c(u) \le L$ **then**
14. $\quad\quad\quad$ **if** $r(u)/c(u) > rc_{max}$ **then**
15. $\quad\quad\quad\quad rc_{max} \leftarrow r(u)/c(u)$
16. $\quad\quad\quad\quad u_{max} \leftarrow u$
17. $\quad\quad\quad$ **end**
18. $\quad\quad$ **else**
19. $\quad\quad\quad U \leftarrow U \setminus \{u\}$
20. $\quad\quad$ **end**
21. $\quad$ **end**
22. $\quad U \leftarrow U \setminus \{u_{max}\}; G^{cur} \leftarrow G^{cur} \setminus \{u_{max}\}$
23. $\quad A_1 \leftarrow A_1 \cup \{u_{max}\}$
24. **until** $U = \emptyset$;
25. $\mathcal{D}_1 \leftarrow \mathsf{DAG}(G \setminus A_1, I), \mathcal{D}_2 \leftarrow \mathsf{DAG}(G \setminus v_{max}, I)$
26. **return** A_1 if $\mathsf{EstInf}(\mathcal{D}_1, I) < \mathsf{EstInf}(\mathcal{D}_2, I)$ else v_{max}

algorithm starts by constructing the DAG graph from original graph G (line 1). Then, it selects the node that maximizes $r(\cdot)$ with the cost less than the budget L is performed (line 4). From line 5 to 24, the greedy phase is showed. The set vertices of $\mathcal{D}$ except I is considered as candidate set in order to choose the solution (line 3). For the repeat loop (lines 6–23), the node that has the ratio of propagation role per cost in current DAG is selected in each iteration. Note that, the current DAG is built from residual

graph. The nodes has cost more than the remaining costs are remove from candidate set U (line 19). This phase stops when the candidate is empty (line 24) or no node has a cost that is less than the remaining cost (line 8). Finally, we approximate the influence of I on current graph by its influence on $\mathcal{D}$,

$$\sigma(I) \approx \mathsf{EstInf}(\mathcal{D}, I) = \sum_{u \in \mathcal{D}} f_{in}(u) \tag{33}$$

Algorithm compares the influence of source when remove set A_1 and v_{max} then returns the solution with the smaller propagation influence (line 26).

5.3 Running time

Next, we consider the worst-case running time of PR-DAG. Suppose that n_θ, m_θ is the largest number of nodes and edges in $\mathcal{D}$. Calculating $MIOA(G, I, d, \theta)$ from G takes $\mathcal{O}(m_\theta + n_\theta \log n_\theta)$, so constructing $\mathcal{D}$ takes $\mathcal{O}(m_\theta + n_\theta \log n_\theta)$ (line 1). Calculate propagation role of each node $u \in \mathcal{D}$ using Algorithm 8 takes $\mathcal{O}(n_\theta + m_\theta)$ time complexity (line 2). For the repeat loop from lines 6 to 32, denote k_1 is the number of iterators. For each iterator, we have to construct DAG and calculate the propagation role of each node. This work can be done in $\mathcal{O}(n_\theta + m_\theta) + \mathcal{O}(n_\theta + n_\theta \log n_\theta) = \mathcal{O}(m_\theta + n_\theta \log n_\theta)$. The selection of nodes that maximize ratio propagation role per cost can be done in linear time. Hence, the total time of loop is $\mathcal{O}(k_1(m_\theta + n_\theta \log n_\theta))$. This is also the complexity of the algorithm.

6 Experiments

In this section, we are going to experimentally evaluate and compare the performance of our proposed algorithms to several widely used methods in the following aspects: (1) objective of solution, and (2) running time on various real social network datasets.

6.1 Experiment settings

We have implemented the proposed algorithms in Python 2.7. All experiments are conducted using a 2 × Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30 GHz with 8 × 16 GB DIMM ECC DDR4 @ 2400 MHz main memory.

Baseline compared For a comparative analysis of the performance of our proposed algorithms, we will compare our proposed algorithms SG, PR-DAG to various baseline algorithms.

- Random: Randomly select nodes within budget L among the $N_d(S)$.
- Degree Centrality (DC): The heuristic algorithm base centrality measure. We select nodes with the highest degree among the $N_d(S)$ and we keep on adding the highest-degree nodes until total costs of the selection of nodes exceed L.

Table 2 Datasets

Dataset	Gnutella Leskovec et al. (2007)	Oregon Leskovec et al. (2005)	Epinions Richardson et al. (2003)	EU email Leskovec et al. (2007)
Num. of nodes	6301	10,670	75,879	265,214
Num. of edges	20,777	22,002	508,837	420,045
Type	Directed	Undirected	Directed	Directed
Avg. degree	3.29	2.06	6.70	1.58

6.1.1 Datasets

We run our experiment on multiple real datasets. In addition, we try to pick datasets of various sizes from different domains, in which the MMR problem is especially applicable. Table 2 shows datasets we used.

Gnutella The snapshot of the Gnutella peer-to-peer file sharing network in August 2002. Nodes represent hosts in the Gnutella network topology and edges represent connections between the Gnutella hosts (Leskovec et al. 2007). It contains 20,777 links among 6,301 hosts.

Oregon This is a graph of Autonomous Systems (AS) peering information inferred from Oregon route-views in March 31 2001 and May 26 2001 (Leskovec et al. 2005).

Epinions The Epinions dataset was extracted (Richardson et al. 2003) and obtained from http://snap.stanford.edu/data. This is a who-trust-whom online social network of a a general consumer review site Epinions.com. Members of the site can decide whether to "trust" each other. All the trust relationships interact and form the Web of Trust which is then combined with review ratings to determine which reviews are shown to the user.

EU Email The network was generated using email data from a large European research institution. For a period from October 2003 to May 2005 (18 months) we have anonymized information about all incoming and outgoing emails of the research institution. Given a set of email messages, each node corresponds to an email address. There is a directed edge between nodes i and j, if i sends at least one message to j.

6.1.2 Parameter settings

Calculating the edge weights We assign the weights of edges in LT model according to previous studies (Goyal et al. 2012; Chen et al. 2010b; Khalil et al. 2014; Kempe et al. 2003; Zhang et al. 2016b). The weight of the edge (u, v) is calculated as follows:

$$w(u, v) = \frac{1}{d_-(v)} \tag{34}$$

Choice of Source nodes We use the methods in Nguyen et al. (2013); Zhang et al. (2016a) to select the source of misinformation. Accordingly, the nodes generating misinformation often have a small number of out-neighbors but are typically located in

the vicinity of celebrities (high-degree nodes). Therefore, sources are chosen randomly from the neighbors of high-degree nodes. For each dataset, we choose a set of source nodes with size $|S| = 100$.

In all experiments, we choose different θ for each different social network. Cha et al. (2009) showed that information mostly propagates within 2 to 5 hops so we choose $d = 3, 4, 5$. To get the expected benefits after removing the set of nodes, we run Mote–Carlo simulation 10,000 times and take the average.

6.2 Experiment results

6.2.1 Comparison of solution quality of MMR *in the case of general cost*

In this experiment, we compare algorithms when d varies, the budget $L = \{10, 15, 25, 40, 60, 80, 100\}$ and the costs of node are uniformly distributed in[1.0, 3.0]. Figures 5, 6, 7, and 8 show the performance of SG, PR-DAG, DC and Random algorithms on Oregon, Epinion, Gnutella and EU Email networks. On avarage, we observe that SG and PR-DAG give better performance compare to the others.

Compare PR-DAG with SG: for Oregon network, PR-DAG is better 1.37 to 12.6% than the result of SG. For Epinions, SG is better than the result of PR-DAG. There is not much diffirent between SG and PR-DAG when $d = 3, 4$ (2.6 and 4.8% times better). In case of $d = 5$, the distance is enlarged. For Gnutella, PR-DAG is better than SG when $d = 3$ and worse than SG when $d = 4, 5$. In EU Email, the running time of SG is over 72 h, so we ignore SG on above dataset. However, as shown in Fig. 8, PR-DAG still performs better than the others. This shows that for large networks (with 265K nodes and 420K edges), PR-DAG finshed within allowed time and gives good performance. Comparing with DC algorithm, the objective function of PR-DAG and SG is 1.05 to 1.72 times better than DC. As the cost of selection of nodes increases, the gap between PR-DAG and the others is getting larger. Random algorithm has a much worse results, indicating that a careful selection solution is critical to block misinformation propagation.

6.2.2 Comparison of solution quality of MMR *in the case of unit-cost version*

To observe more clearly on the performance of these algorithms, we have conducted experiments with the case of all node costs are equal to 1 on datasets. We set L varies from 1 to 50, and d varies from 3 to 5. Figures 9, 10 and 11 display the objective function on Gnutella, Oregon and Epinions, respectively. PR-DAG is very close to SG. The larger value of k, the smaller the gap betwen PR-DAG and SG. Especially, with cost greater than 25, PR-DAG and the SG achieve the same level of performance. It is very clear to prove that PR-DAG can find the solution during the constructing DAG from original graph which effectively approximate the influence between nodes. Like the previous test case, for EU Email, the running time of SG is over 72 h, so we ignore SG on above dataset. Therefore, SG algorithm is not feasible when finding the solution on large networks. Comparing PR-DAG with DC, PR-DAG outperforms DC algorithm in term of benefit of selected nodes. On avarage, PR-DAG is 1.51 times

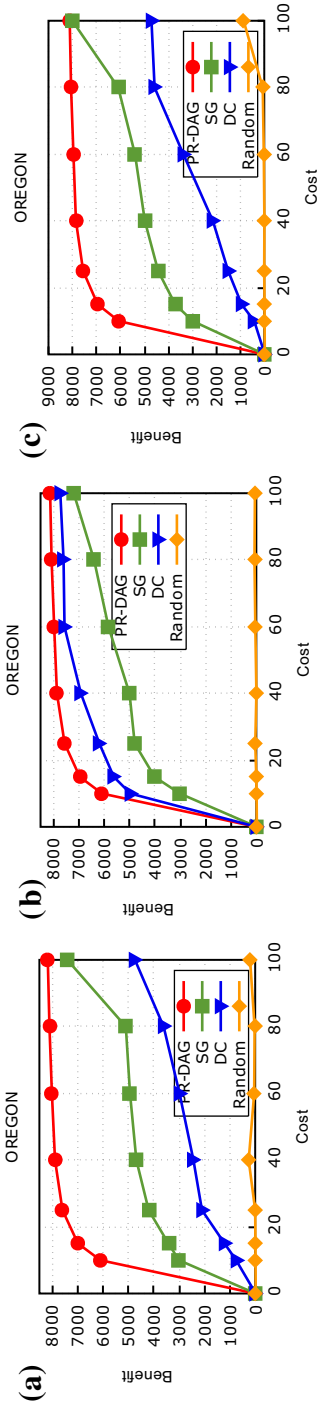

Fig. 5 Comparison on MMR problem for Oregon with general cost and $d = 3, 4, 5$, **a** $d = 3$, **b** $d = 4$, **c** $d = 5$

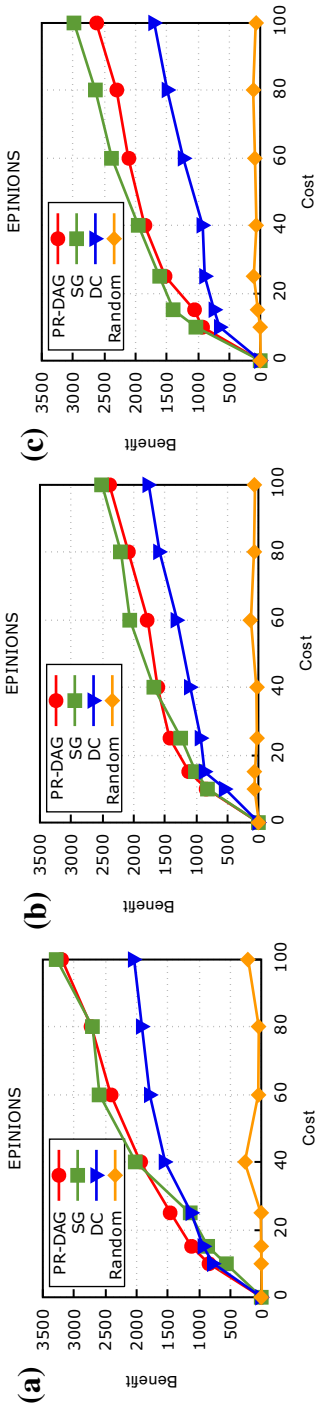

Fig. 6 Comparison on MMR problem for Epinions with general cost and $d = 3, 4, 5$, **a** $d = 3$, **b** $d = 4$, **c** $d = 5$

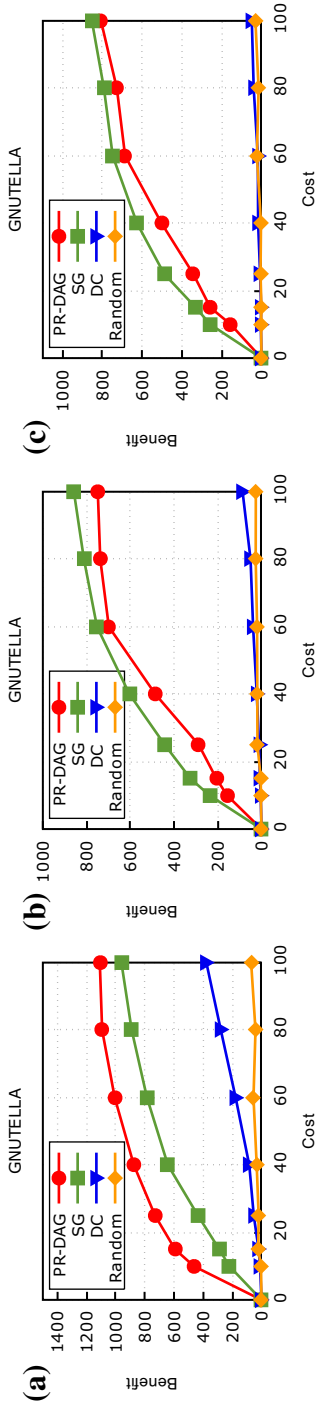

Fig. 7 Comparison on MMR problem for Gnutella with general cost and $d = 3, 4, 5$, **a** $d = 3$, **b** $d = 4$, **c** $d = 5$

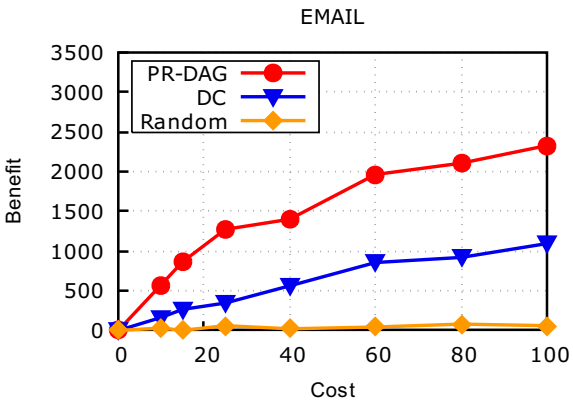

Fig. 8 Comparison on MMR problem for EU Email with general cost and $d = 3$

better on Gnutella, 5.12 times better on Oregon, 1.43 times better on Epinions, and 1.14 times better on EU Email. These results are also consistent with what have been observed in the previous test case.

6.2.3 Scalability

In this subsection, we show some running time results to demonstrate scalability. Tables 3 and 4 show the running time for PR-DAG and SG. We did not show the running time of Random and DC because they are fast heuristics that finish in the order of seconds. As expected, PR-DAG is much faster than SG. In average, PR-DAG is up to 32.5 and 45.4 times faster SG on general cost and unit-cost case, respectively. We also show detail the running times of PR-DAG and SG on Oregon in Figs. 12 and 13, for other networks we got similar results. We observe that as L increases, the gap between PR-DAG and SG increases which is consistent with their analysis theoretical complexities. For the large-scale dataset EU Email, PR-DAG took less than 11 h to select the set nodes, while SG could not finish in 72 h. This again emphasizes the scalability of PR-DAG for MMR on large-scale networks.

7 Conclusion

In this paper, we investigated a new NP-hard problem of Maximizing Misinformation Restriction with the aim of maximizing the influence reduction of misinformation source on OSNs by removing the set of nodes within time constraints and the limited budget. We proved the hardness results, and provided the approximation algorithm to solve the problem. In addition, due to the high runtime complexity of this approximation algorithm, a much more efficient heuristic algorithm was also proposed, named PR-DAG. This algorithm shows large performance advantage in both solution quality and running time on real social networks.

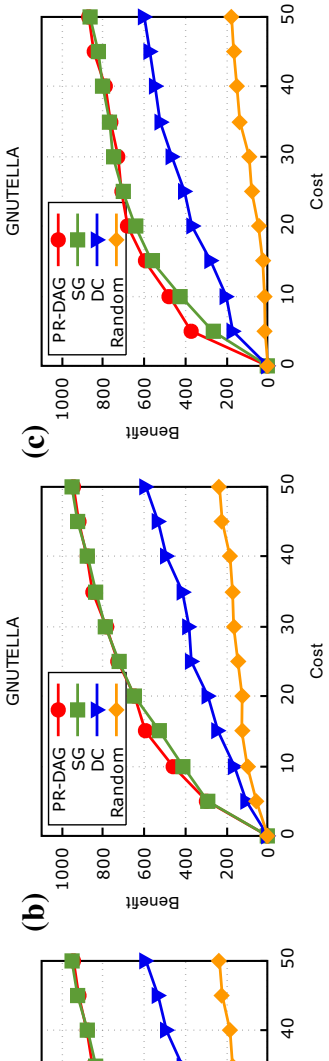

Fig. 9 Comparison on unit-cost version of MMR problem for Gnutella. **a** $d = 3$, **b** $d = 4$, **c** $d = 5$

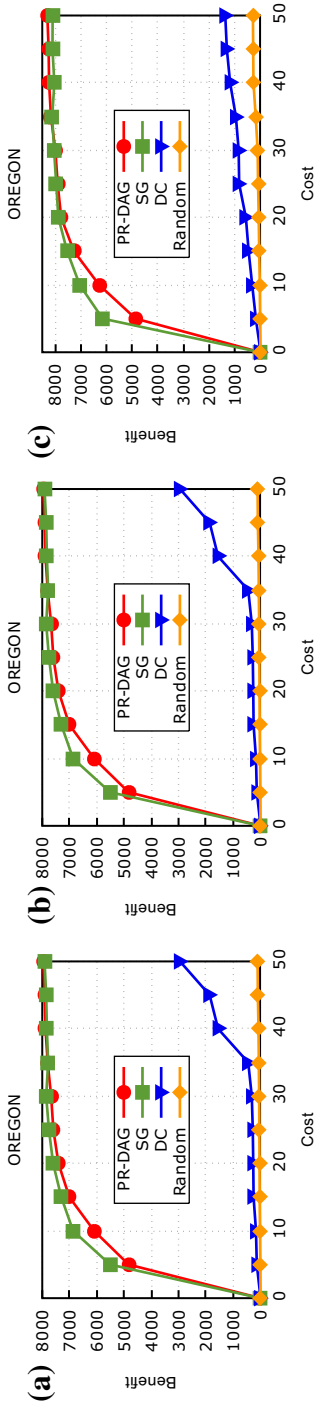

Fig. 10 Comparison on unit-cost version of MMR problem for Oregon. **a** $d = 3$, **b** $d = 4$, **c** $d = 5$

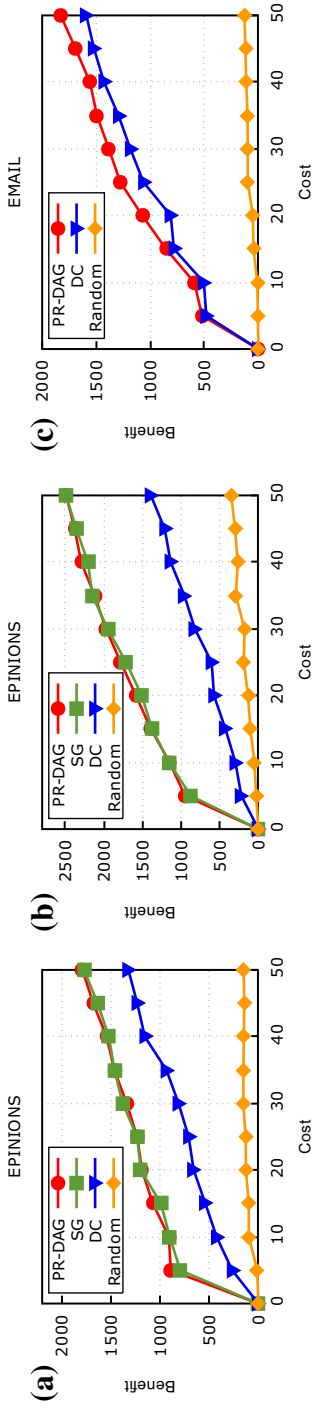

Fig. 11 Comparison on unt-icost version of MMR problem for Epinion and Email, **a** Epinions, $d = 3$, **b** Epinions, $d = 4$, **c** Email, $d = 3$

Table 3 Running time (s) of PR-DAG and SG in general cost version when $L = 100$

Dataset	$d = 3$		$d = 4$		$d = 5$	
	PR-DAG	SG	PR-DAG	SG	PR-DAG	SG
Oregon	800.30	20,556.32	880.92	26,585.70	839.97	27,290.34
Epinions	9255.00	18,421.07	10,084.91	24,359.07	9984.81	26,665.14
Gnutella	172.53	1152.47	440.92	1721.73	676.95	1996.49
EU Email	19,973.13	–				

Runs terminated when running time $t > 72$ h (shown by '–')

Table 4 Running time (s) of PR-DAG and SG in unit-cost version when $L = 50$

Dataset	$d = 3$		$d = 4$		$d = 5$	
	PR-DAG	SG	PR-DAG	SG	PR-DAG	SG
Gnutella	98.79	999.78	252.23	1442.99	386.64	1702.31
Oregon	584.58	22,525.60	574.98	24,931.68	588.10	26,713.52
Epinions	27,482.42	65,809.16	28,858.84	90,167.34		
EU Email	39,014.17	–				

Runs terminated when running time $t > 72$ h (shown by '–')

Appendix

We let $e = (u, v)$, $f = (u', v')$, $\mathcal{G}^e(G \setminus X)$ is the set of sample graph where incoming edge $e = (u, v)$ is selected for node v, $\mathcal{G}^{\overline{e}}(G \setminus X)$ is the set of sample graph where a different incoming edge $\overline{e} = (y, v)$ is selected for node v and $\mathcal{G}^{\emptyset}(G \setminus X)$ is the set of sample graph where no incoming edge $\overline{e} = (y, v)$ is selected for node v. According to Khalil et al. (2014), we have the following results:

Proposition 2 (Khalil et al. (2014), proposition 1) *For every live-edge* $g \in \mathcal{G}^{\emptyset}(G \setminus X)$, *there exits a corresponding live-edge graph* $\widetilde{g} \in \mathcal{G}^e(G \setminus X)$ *and vice versa. If* $g = (V, E_g)$ *then* $\widetilde{g} = (V, E_g \cup \{e\})$.

Proposition 3 (Khalil et al. (2014), proposition 2) $\mathcal{G}(G \setminus (X \cup \{e\})) \subseteq \mathcal{G}(G \setminus X)$ *and furthermore* $\mathcal{G}(G \setminus (X \cup \{e\})) = \mathcal{G}^{\overline{e}}(G \setminus X) \cup \mathcal{G}^{\emptyset}(G \setminus X)$.

Proposition 4 (Khalil et al. (2014), proposition 3) *Given* $f = (u', v') \in E \setminus X$, $v' \neq v$, *let* $t = |\mathcal{G}^{\emptyset}(G \setminus (X \cup \{f\}))|$ *then* $\mathcal{G}^{\emptyset}(G \setminus X)$ *can be partitioned into* t *sets* $\{\Phi_i\}_{i=1}^{t}$ *such that, for every* Φ_i *there exits a corresponding* $g_i \in \mathcal{G}^{\emptyset}(G \setminus (X \cup \{f\}))$ *and vice versa.*

Proposition 5 (Khalil et al. (2014), proposition 4) *For every* $\Phi_i \subseteq \mathcal{G}^{\emptyset}(G \setminus X)$ *and its associated* $g_i \in \mathcal{G}^{\emptyset}(G \setminus (X \cup \{e\}))$, $\Pr[g_i | G \setminus (X \cup \{f\})] = \sum_{H \in \Phi_i} \Pr[H | G \setminus X]$.

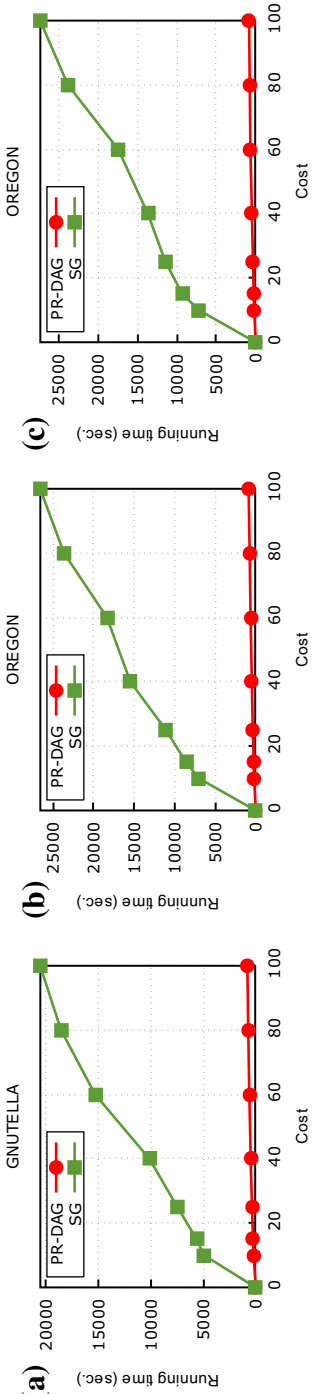

Fig. 12 Comparison on MMR problem for Oregon with general node costs for $d = 3, 4, 5$, **a** $d = 3$, **b** $d = 4$, **c** $d = 5$

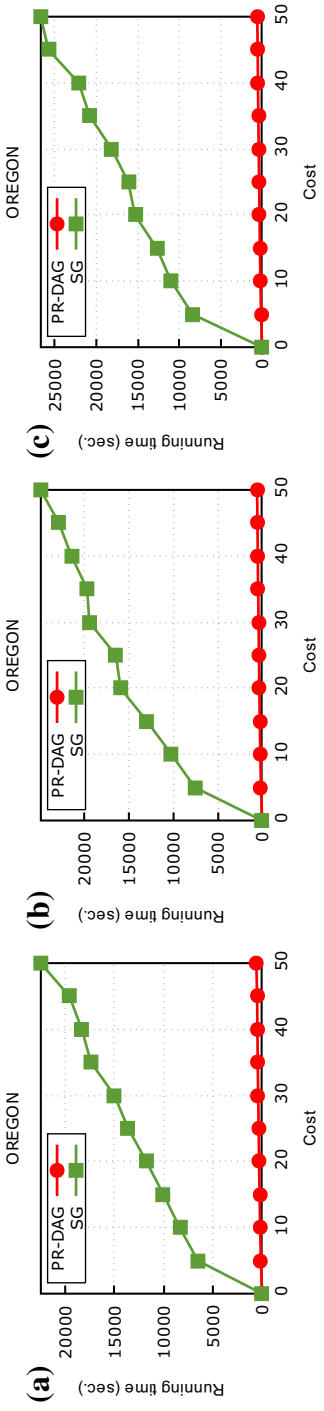

Fig. 13 Running time of PR-DAG and SG on Oregon with unit-costs and $d = 3, 4, 5$, **a** $d = 3$, **b** $d = 4$, **c** $d = 5$

Proof of Lemma 3 We need to show that $\sigma_{d,E}(S, X) \geq \sigma_{d,E}(S, X \cup \{e\})$. The idea of the proof is similar to the theorem 5 in Khalil et al. (2014). Using Proposition 2 and 3, we have:

$$\sigma_{d,E}(S, X) - \sigma_{d,E}(S, X \cup \{e\}) = \sum_{g \in \mathbb{G}(G \setminus X)} \Pr[g|G \setminus X] f_d(g, S)$$

$$- \sum_{g \in \mathcal{G}(G \setminus (X \cup \{e\}))} \Pr[g|G \setminus (X \cup \{e\})] f_d(g, S) = \sum_{g \in \mathcal{G}^e(G \setminus X)} \Pr[g|G \setminus X] \cdot f_d(g, S)$$

$$+ \sum_{g \in \mathcal{G}^{\emptyset}(G \setminus X)} \Big(\Pr[g|G \setminus X)] - \Pr[g|G \setminus (X \cup \{e\})] \Big) \cdot f_d(g, S)$$

$$+ \sum_{g \in \mathcal{G}^{\overline{e}}(G \setminus X)} \Big(\Pr[g|G \setminus X)] - \Pr[g|G \setminus (X \cup \{e\})] \Big) \cdot f_d(g, S)$$

Recall that $e = (u, v)$, for $g \in \mathcal{G}^{\emptyset}(G \setminus X)$, we have:

$$\Pr[g|G \setminus X] - \Pr[g|G \setminus (X \cup \{e\})] = -w(u, v) \prod_{v' \neq v} P(v', g, G \setminus X) \tag{35}$$

For $g \in \mathcal{G}^{\overline{e}}(G \setminus X)$ we have $P(v, g, G \setminus X) = P(v, g, G \setminus (X \cup \{e\})) = w(\overline{e})$ which leads to $\Pr[g|G \setminus X] = \Pr[g|G \setminus (X \cup \{e\})]$, it infers:

$$\sigma_{d,E}(S, X) - \sigma_{d,E}(S, X \cup \{e\}) = \sum_{g \in \mathcal{G}^e(G \setminus X)} \Pr[g|G \setminus X] \cdot f_d(g, S)$$

$$+ \sum_{g \in \mathcal{G}^{\emptyset}(G \setminus X)} -w(u, v) \prod_{v' \neq v} p(v', g, G \setminus X) \cdot f_d(g, S)$$

Using prop. 2, for $g \in \mathcal{G}^{\emptyset}(G \setminus X)$ there exits a corresponding $\widetilde{g} \in \mathcal{G}^e(G \setminus X)$ and $\Pr[\widetilde{g}|G \setminus X] = w(u, v) \prod_{v' \neq v} p(v', \widetilde{g}, G \setminus X)$. Therefore,

$$\sigma_{d,E}(S, X) - \sigma_{d,E}(S, X \cup \{e\}) = \sum_{g \in \mathcal{G}^{\emptyset}(G \setminus X)} \Pr[\widetilde{g}|G \setminus X] \Big(f_d(\widetilde{g}, S) - f_d(g, S) \Big) \tag{36}$$

We can see that g is a subgraph of $\widetilde{g}$, the set of vertices which can reach from S in g is subset of the set of vertices which can reach from S in $\widetilde{g}$. Hence, $f_d(\widetilde{g}, S) - f_d(g, S) \geq 0$, which completes the proof. □

Proof of Lemma 4 the idea of the proof is similar to that of theorem 6 in Khalil et al. (2014). For edge $f \in E$, let $t = |\mathcal{G}(G \setminus (X\{f\}))|$. From Proposition 4, we can partition $\mathcal{G}^{\emptyset}(G \setminus X)$ into t sets $\{\Phi\}_{i=1}^{t}$, rewrite (36) as:

$$\sigma_{d,E}(S, X) - \sigma_{d,E}(S, X \cup \{e\}) = \sum_{g \in \mathcal{G}^{\emptyset}(G \setminus X)} \Pr[\widetilde{g}|G \setminus X]\Big(f_d(\widetilde{g}, S) - f_d(g, S)\Big)$$
$$= \sum_{i=1}^{t} \sum_{g \in \Phi_i} \Pr[\widetilde{g}|G \setminus X]\Big(f_d(\widetilde{g}, S) - f_d(g, S)\Big) \tag{37}$$

Using similar reasoning to that in Eq. (36) in the proof of lemma 1 for $G \setminus (X \cup \{f\})$, we have:

$$\sigma_{d,E}(S, X \cup \{f\}) - \sigma_{d,E}(S, X \cup \{f, e\})$$
$$= \sum_{g \in \mathcal{G}^{\emptyset}(G \setminus (X \cup \{f\}))} \Pr[\widetilde{g}|G \setminus (X \cup \{f\})]\Big(f_d(\widetilde{g}, S) - f_d(g, S)\Big) \tag{38}$$

We will compare two Eqs. (37) and (38) term by term for each $g_i \in \mathcal{G}^{\emptyset}(G \setminus X)$, $i = 1, \ldots, t$. It can be divided into two cases: (1) $\Phi_i = \{X_i\}$ in case in g_i has another incoming edge to v' not f, now the terms in two equations are equal; (2) in the case in g_i has only incoming edge to v' is f, then $\Phi_i = \{g_i, g_i'\}$, we need to prove:

$$\Pr[\widetilde{g_i}|G \setminus X]\Big(f_d(\widetilde{g_i}, S) - f_d(g_i, S)\Big) + \Pr[\widetilde{g_i'}|G \setminus X]\Big(f_d(\widetilde{g_i'}, S) - f_d(g_i', S)\Big)$$
$$\geq \Pr[\widetilde{g_i}|G \setminus (X \cup \{f\})]\Big(f_d(\widetilde{g_i}, S) - f_d(g_i, S)\Big) \tag{39}$$

Using prop. 5 in Khalil et al. (2014), we have:

$$\Pr[\widetilde{g_i}|G \setminus (X \cup \{f\})] = \Pr[\widetilde{g_i}|G \setminus X] + \Pr[\widetilde{g_i'}|G \setminus X] \tag{40}$$

Hence, inequality (39) is true if:

$$f_d(\widetilde{g_i'}, S) - f_d(g_i', S) \geq f_d(\widetilde{g_i}, S) - f_d(g_i, S) \tag{41}$$

Note that $\widetilde{g_i'} = (V, E_{g_i} \cup \{f\})$ and live-edge graphs are constructed in a way that each node has at most one incoming edge. We can see that: a reachability path in $\widetilde{g_i}$ is clearly presented in $\widetilde{g_i'}$, hence if removing edge e from $\widetilde{g_i}$ results in unreachability of some nodes in g_i. Similarly, some nodes become unreachable when removing e from $\widetilde{g_i'}$. Removing edge e from $\widetilde{g_i'}$ may disconnect some additional nodes whose paths derived from the source including edge f. Therefore, the reduction in reachable nodes when removing edge e from $\widetilde{g_i'}$ is the same or larger than the reduction when removing e from $\widetilde{g_i}$, it implies (41) is true. □

References

Bhagat S, Goyal A, Lakshmanan LV (2012) Maximizing product adoption in social networks. In: Proceedings of the fifth ACM international conference on Web search and data mining, Seattle, Washington, pp 603–612

Budak C, Agrawal D, Abbadi AE (2011) Limiting the spread of misinformation in social networks, In: Proceedings of the 20th international conference on world wide web, WWW '11, ACM, New York, NY. https://doi.org/10.1145/1963405.1963499

Cha M, Mislove A, Gummadi KP (2009) A measurement-driven analysis of information propagation in the Flickr social network. In: Proceedings of the 18th international conference on world wide web, New York, USA, pp 721–730

Chen W, Lakshmanan LVS, Castillo C (2013) Information and influence propagation in social networks. Morgan and Claypool, San Rafael

Chen W, Lu W, Zang N (2012) Time-critical influence maximization in social networks with time-delayed diffusion process. In: Proceedings of the twenty-sixth AAAI conference on artificial intelligence, Toronto, Ontario, pp 592–598

Chen W, Wang C, 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, Washington, DC, pp 1029–1038

Chen W, Wang C, Wang Y (2010) Scalable influence maximization in social networks under the linear threshold model. In: Proceedings of the 2010 IEEE international conference on data mining, Washington, pp 88–97

Domingos P, Richardson M (2001) Mining the network value of customers. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining, New York, USA, pp 57–66

Domm P (2013) False rumor of explosion at white house causes stocks to briefly plunge, AP Confirms Its Twitter Feed Was Hacked CNBC. http://www.cnbc.com/id/100646197(2013). Accessed 23 April 2013

Gentzkow M (2017) Social media and fake news in the 2016 election. Stanford Web. http://news.stanford.edu/2017/01/18/stanford-study-examines-fake-news-2016-presidential-election/ (2017). Accessed 24 June 2017

Goyal A, Lu W, Lakshmanan LVS (2012) SIMPATH: an efficient algorithm for influence maximization under the linear threshold model. In: Proceeding IEEE 11th international conference on data mining, Vancouver, BC, pp 211–220

He X, Song G, Chen W, Jiang Q (2011) Influence blocking maximization in social networks under the competitive linear threshold model technical report, CoRR abs/1110.4723

Hughes AL, Palen L (2009) Twitter adoption and use in mass convergence and emergency events. Int J Emerg Manage 6(3):248–260

Kempe D, Kleinberg J, 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. https://doi.org/10.1145/956750.956769

Khalil EB, Dilkina B,Song L (2014) Scalable diffusion-aware optimization of network topology. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, New York, pp 1226–1235

Khuller S, Moss A, Naor JS (1999) The budgeted maximum coverage problem. Inf Process Lett 70(1):39–45

Kimura M, Saito K, Motoda H (2008) Solving the contamination minimization problem on networks for the linear threshold model. In: Pacific rim international conference on artificial intelligence. https://doi.org/10.1007/978-3-540-89197-0_94

Kimura M, Saito K, Motoda H (2009) Blocking links to minimize contamination spread in a social network. In: ACM transactions on knowledge discovery from data. https://doi.org/10.1145/1514888.1514892

Kottasov I (2017) Facebook targets 30,000 fake accounts in France. CNN media Web. http://money.cnn.com/2017/04/14/media/facebook-fake-news-france-election/index.html. Accessed 24 June 2017

Kwon S, Cha M, Jung K, Chen W, and Wang Y (2013) Prominent features of rumor propagation in online social media. In: Proceeding of IEEE 13th international conference on data mining. https://doi.org/10.1109/ICDM.2013.61

Leskovec J, Adamic LA, Huberman BA (2007) The dynamics of viral marketing. ACM Trans Web. https://doi.org/10.1145/1232722

Leskovec J, Kleinberg J, Faloutsos C (2005) Graphs over time: densification laws, shrinking diameters and possible explanations. In: Proceeding of ACM SIGKDD international conference on knowledge discovery and data mining. https://doi.org/10.1145/1081870.1081893

Leskovec J, Kleinberg J, Faloutsos C (2007) Graph evolution: densification and shrinking diameters. ACM Trans Knowl Discov Data 1(1):2

Liu B, Cong G, Xu D, Zheng Y (2012) Time constrained influence maximization in social networks. In: Proceeding of IEEE 12th international conference on data mining, Belgium, Brussels, pp 439–448

Nguyen H, Zheng R (2013) On budgeted influence maximization in social networks. IEEE J Sel Areas Commun 31(6):1084–1094

Nguyen NP, Yan G, Thai MT (2013) Analysis of misinformation containment in online social networks. Comput Netw 57:21332146

Qazvinian V, Rosengren E, Radev DR, Mei Q (2011) Rumor has it: identifying misinformation in microblogs. In: Proceedings of the conference on empirical methods in natural language processing, Edinburgh, pp 1589–1599

Richardson M, Agrawal R, Domingos P (2003)Trust management for the semantic web. In: Proceeding of international semantic web conference. https://doi.org/10.1007/978-3-540-39718-2_23

Sutter JD (2017) How bin Laden news spread on Twitter. CNN Web. http://edition.cnn.com/2011/TECH/social.media/05/02/osama.bin.laden.twitter/index.html. Accessed 23 June 2017

Valiant LG (1979) The complexity of enumeration and reliability problems. SIAM J Comput 8(3):410–421

Wolfsfeld G, Segev E, Sheafer T (2013) Social media and the Arab Spring: politics comes first. Int J Press Polit 18(2):115–137

Yadron D (2017) Twitter deletes 125,000 Isis accounts and expands anti-terror teams. The Guardian Web. https://www.theguardian.com/technology/2016/feb/05/twitter-deletes-isis-accounts-terrorism-online. Accessed 24 June 2017

Zhang H, Alim M, Li X, My TT, Nguyen H (2016a) Misinformation in online social networks: catch them all with limited budget. ACM Trans Inf Syst 34(3):18

Zhang Y, Adigay A, Saha S, Vullikanti A, Prakash A (2016b) Near-optimal algorithms for controlling propagation at group scale on networks. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2016.2605088

Zhang H, Dinh TN, Thai MT (2013) Maximizing the spread of positive influence in online social networks. In: Proceeding IEEE 33rd international conference on distributed computing systems, Philadelphia, PA, pp 317-326

Zhang H, Kuhnle A, Zhang H, Thai MT (2016c) Detecting misinformation in online social networks before it is too late. In: Proceeding IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM). https://doi.org/10.1109/ASONAM.2016.7752288

Zhang Y, Prakash B (2015) Data-aware vaccine allocation over large networks. ACM Trans Knowl Discov Data. https://doi.org/10.1145/2803176

Zhang Y, Prakash BA (2014) Scalable vaccine distribution in large graphs given uncertain data. In: Proceeding of the ACM international conference on information and knowledge management, Shanghai, pp 1719–1728