TRANSACTIONS ON EMERGING TELECOMMUNICATIONS TECHNOLOGIES
Trans. Emerging Tel. Tech. 2017;**28**:e3054.
Published online 28 June 2016 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/ett.3054

RESEARCH ARTICLE

Influence maximization by probing partial communities in dynamic online social networks

Meng Han[1], Mingyuan Yan[2], Zhipeng Cai[1], Yingshu Li[1]*, Xingquan Cai[3] and Jiguo Yu[4]

[1] The Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA
[2] Department of Computer Science and Information Systems, University of North Georgia, Dahlonega, GA 30597, USA
[3] North China University of Technology, College of Information Engineering, Beijing, China
[4] School of Information Science and Engineering, Qufu Normal University, Rizhao, Shandong 276826, China

ABSTRACT

With the rapid development of online social networks, exploring influence maximization for product publicity and advertisement marketing has attracted strong interests from both academia and industry. However, because of the continuous change of network topology, updating the variation of an entire network moment by moment is resource intensive and often insurmountable. On the other hand, the classical influence maximization models *Independent Cascade* and *Linear Threshold* together with their derived varieties are all computationally intensive. Thus, developing a solution for dynamic networks with lower cost and higher accuracy is in an urgent necessity. In this paper, a practical framework is proposed by only probing partial communities to explore the real changes of a network. Our framework minimizes the possible difference between the observed topology and the real network through several representative communities. Based on the framework, an algorithm that takes full advantage of our divide-and-conquer strategy, which reduces the computational overhead, is proposed. The systemically theoretical analysis shows that the proposed effective algorithm could achieve provable approximation guarantees. Empirical studies on synthetic and real large-scale social networks demonstrate that our framework has better practicality compared with most existing works and provides a regulatory mechanism for enhancing influence maximization.

***Correspondence**
Yingshu Li, The Department of Computer Science, Georgia State University, Atlanta, GA 30597, USA.
E-mail: zcai@gsu.edu

Received 30 October 2015; Revised 20 April 2016; Accepted 15 May 2016

1. INTRODUCTION

Social influence is the phenomenon that one's emotions, opinions or behaviours can induce his or her friends to behave in a similar way. This phenomenon has become a popular topic recently because of its unlimited potential in business and marketing. For instance, The Alibaba Group, one of China's E-commerce giants, reported that the value of its online transactions during China's unofficial 'Singles' Day'holiday had hit $9.3bn. On the same day in the previous year, the total purchases through Alibaba was $5.75bn.[†] While in E-commerce environments (also known as one type of Online Social Networks (*OSNs*)), strong virtual relationships existing among users construct the basic structure in modern social life. Substantial commercial opportunities are coming with lots of social influence based applications. From this aspect, how to maximize the influence on potential customers is one of the most important keys to open the door of the unlimited business opportunities [1, 2], which drives the extensive investigation on the influence maximization in social networks.

Influence maximization is based on trust among individuals' within close social circle, such as families, friends and co-workers. Word-of-mouth or viral marketing differentiates [3] itself from other marketing strategies and is widely supported by lots of research and industrial applications. Besides the strategy of distributing promoted samples, maximization of influence on the social network could also be applied to other similar services [4, 5] not only limited by trial devices but also include marketing for releasing new music (Spotify.com), cloud computing resource (AWS from Amazon.com) and even financial services (www.usaa.com/). Companies, such as Spotify, Amazon or USAA, mainly, promote their new services

[†]http://www.nytimes.com/.

through distributing emails with a special offer to solicit new customers. An important issue of these social networks is that they are dynamic and community based. Applying our algorithm could help them to find out who are the optimal target groups that the emails or advertisements should be geared towards.

As one of the most popular research topics in social networks, many existing literatures focusing on the influence maximization in social networks can be found. For example, [6, 7] proposed several models to simulate the influence diffusion process. However, influence maximization in social networks is still an developing problem. Because of the complexity and diversity of the phenomena, researchers are still facing a lot of challenges concerning how to analyse and utilize the influence in *OSNs*. First, OSNs are dynamic networks. Therefore, the change of network topology directly affects the diffusion of influence. In addition, almost all of the classical models such as Independent Cascade (*IC*) and Linear Threshold *(LT)* together with their many derived varieties involving Monte-Carlo simulation are incredibly resource intensive. The basic idea of both *IC* and *LT* is to find a subset $S^* \in V$ such that $|S^*| = k$ and $\delta(S^*) = \max\{\delta(S)||S| = k, S \in V\}$, where the function $\delta()$ is the information propagation strategy. Because the influence maximization problem is unfortunately *NP-hard*, it is impossible to find the most influential nodes in a network. It is even more difficult to pursue the optimal node sets that can maximize the influence in a dynamic social network. Besides all the other challenges, updating a network to reflect its dynamic nature with time is extremely resource consuming in large social networks. Therefore, in this paper, we are interested in proposing an efficient integrated solution to select the most influential nodes in dynamic social networks considering the challenges and features of OSNs.

After a comprehensive background research from real OSNs, we summarize the following facts:

(1) First of all, communication is one of the essential and distinctive features of social networks. Because the nature of a dynamic network concretely comes from each node in each community, the final influence effect could also be evaluated by nodes in each community.
(2) In a whole network, dynamics do not exist in every corner. It mainly comes from several parts of the network instead. The parts commonly reflect in the unit of a community.
(3) Detecting significant communities, which reflect the change of a network and performing selective updates, could save a plenty of computation overhead.

Figure 1 shows an example of the influence diffusion in a dynamic social network. The top figure presents the network at time $t = 0$; the middle figure describes the changing of the network at time $t = 1$, and the bottom figure is the network topology at the end $t = 2$. From left to right, the network is divided into three communities. Through the time flow from top to bottom, we could get that only the two communities (the left and the right in the dashed line frames) have changed their topology. But the nodes in the middle part remain unchanged. From this example, we notice that it would be more efficient if we could identify the two changed communities but ignore the middle community during updating. Therefore, probing the most active communities to approximate the global evolution of the network would be a very effective solution. Most existing research surprisingly ignores the advantages of the community feature. On the other hand, the complexity of solving the influence maximization problem rapidly increases with the size of the network. Therefore, finding influential local nodes in each smaller community could be much more efficient.

It is worth pointing out that the objective of our work is to track the network's global dynamics as well as reducing the cost brought by frequently updating the whole network. We utilize the 'community' instead of 'node' as a unit to probe the change of the network because the community is the basic and natural structure in large networks, which is a better choice compared with the node. Even though the updating unit is the community, the changing of nodes and links among nodes in the communities are more commonly the updating targets. For each iteration, based on our theoretical analysis, when b communities are selected to be actually updated, the nodes and the links among nodes in the selected communities are going to be updated. The reason we do not take a node as the unit to update the network is that from an overall perspective, even the changing of only one node in a network will only result in several relationships changing. Frequently updating the network node by node will bring in more redundant costs because of updating their neighbours. On the other hand, a community will cover several nodes with closer relationships. The observations in previous results [25] show that most of the dynamics in large networks have some kind of local effect [8] confirming the advantage of communities over nodes. The local update of communities could update the dynamic changing within specific areas.

Again, considering the node 'Michael' in the left frame, from time $t = 0$ to $t = 2$, 'Michael' disappears from the original topology. In this case, we could consider it as node 'Michael' has been excluded from the network at time $t = 0$, which is the opposite process of a new node joining. Our algorithm considers the changing of each node within the selected communities. Both nodes joining and leaving are considered in the algorithm. Thus, our algorithm considers the community as a unit to probe the dynamic of networks from a global sense, but the dynamics of nodes is the actual cell being updated.

Although it is not always the case that one influential node in a community corresponds to an influential node in the whole network, apparently, an influential node in one community has a stronger influence based on its degree and neighbours' density compared with normal nodes in the whole network. The remaining challenge is how to fill

Trans. Emerging Tel. Tech. 2017;**28**:e3054. © 2016 John Wiley & Sons, Ltd.
DOI: 10.1002/ett

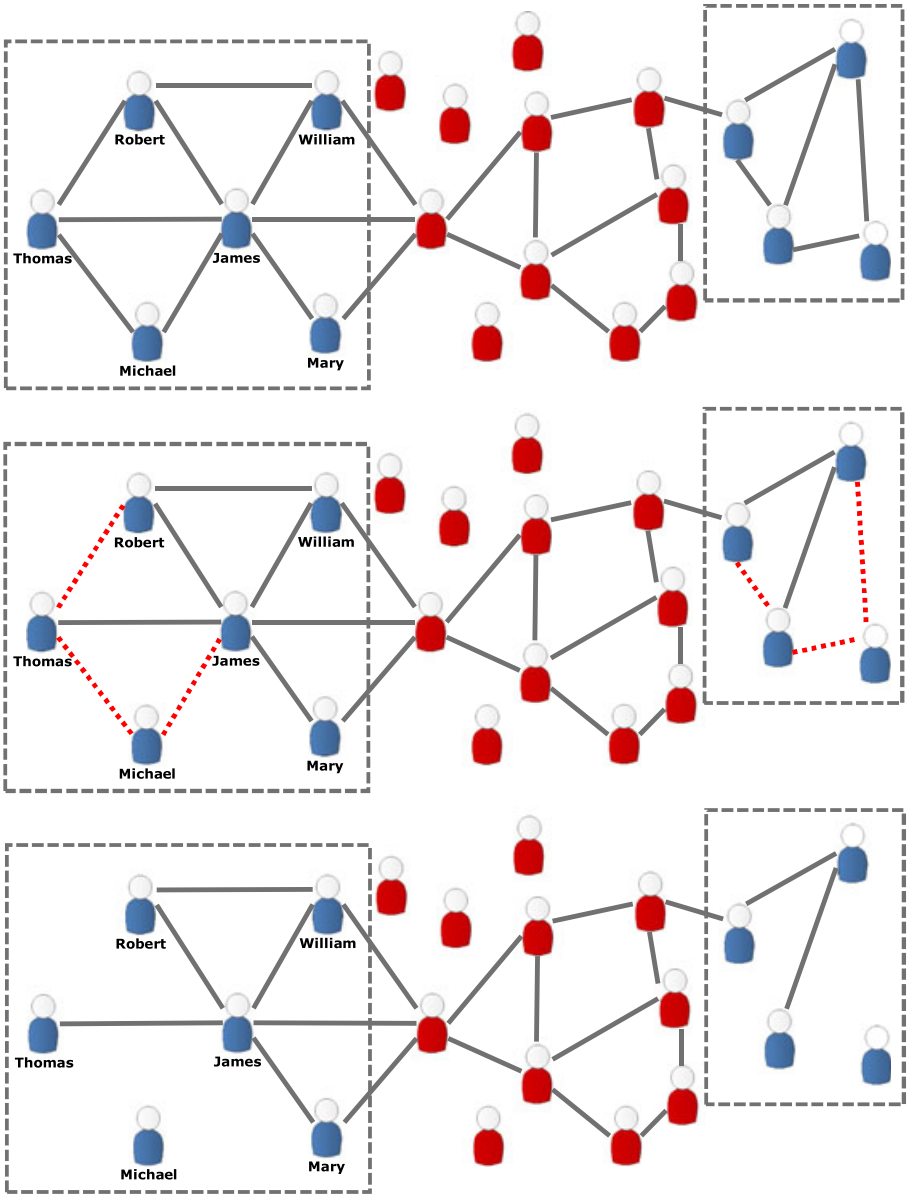

Figure 1. Probing community for dynamic network.

the gap between the local influential nodes in a community and the global influential nodes in a social network. In [9], the authors try to reduce the global computational cost of influence maximization algorithms by splitting the computation cost to many smaller communities. Inspired by [9], we take full advantage of the divide-and-conquer strategy with many differences and improvements. First, the network topology considered in our work is dynamic. Second, the output of our solution could be the maximal influential node set with a specific time stamp instead of only one influential node set in a static social network. From this point, our solution is a superset of [9]. Third, in [9], the considered diffusion speed depends on a static network topology, which is not a general scenario in social networks. Finally, we propose innovative approaches other than using the traditional straightforward method.

Our contributions are summarized as follows:

(1) We formulate and model the influence maximization problem in a dynamic social network. The model explores the potential of employing a community-based algorithm to achieve the influence maximization in dynamic OSNs.
(2) We propose an effective algorithm by probing only part of communities instead of the whole network in each time stamp to save computation cost. Meanwhile, we provide theoretical analysis for the

error bound of the influence diffusion under the defined model.

(3) Based on the network structure derived from the dynamic environment, we develop a community-based dynamic programming algorithm together with multiple optimization techniques of partition and combination processes to mine the influential nodes.

(4) Last but not the least, our experimental results on several real datasets show that in both synthetic and real large networks, our solution clearly improves practicality and accuracy against several existing algorithms.

The rest of the paper is organized as follows. Section 2 reviews the related works. Section 3 presents the preliminaries and problem definition. Then we introduce our proposed model with analysis and the algorithm in Section 4. The evaluation results based on synthetic and real datasets are shown in Section 5. Section 6 concludes the paper.

2. RELATED WORKS

For the influence maximization in OSNs, the *IC* and *LT* models [6] together with their extensions [10] set the foundation for most of the existing cascading algorithms. Because Kempe *et al*. [6] formulated the influence maximization problem as an optimization problem, a series of empirical studies have been performed on influence learning [11–13], algorithm optimization [14–16], scalability promotion [17, 18], influence of group conformity [19] and influence of location exploration [20]. Anadiotis, Christos, *et al*. [21] illustrated how information-centric networks can effectively address practical issues rising in multimedia applications and social networking technologies [22]. In [21], the authors provide certain observation that the influence in social networks are information-centric especially within a community, which motivate us to maximize the influence from the community aspect to the global network. Leskovec *et al*. [7] modelled the outbreak detection problem and proved that the influence maximization problem is a special case of their new problem. In [7], a Cost-Effective Lazy Forward (*CELF*) scheme taking the advantage of submodular property is proposed. It achieved 700 times speedup in selecting seed vertices compared with the classical greedy algorithm [6]. *CELF* set a milestone of using brute force solution to solve the influence maximization problem. However, as indicated in [18], solutions proposed in *CELF* lack of scalability. Therefore, Chen *et al*. [18] proposed several efficiency heuristic algorithms based on the arborescence structure, which is a structure that could handle million-sized graphs. The proposed algorithm spreads influence similar to the way that the greedy algorithm does, while achieving more than six orders of magnitude faster than the greedy algorithm.

Besides literatures focusing on the fundamental influence maximization problem and its several variants mentioned earlier, existing literatures related to our work can be divided into two groups: the dynamic network models and community-based influence maximization.

Esmaeilyfard, Khiadani, *et al*. [23] focused on the idea of establishing a dynamic vehicular network to manage social groups and the information of social applications. Reference [24] addressed the characteristics of high node mobility and high dynamism of network topology. Zhuang *et al*. [25] considered structure changing over a network. They aimed at probing a subset of nodes in a social network to estimate the real influence diffusion process. However, different from us, [25] focuses on investigating a network by probing several nodes with the highest influence increment instead of using the community scope. For a large-scale network, probing and updating several separated nodes are very time-consuming, and it is hard to calculate the increment for each node. In contract, communities as a natural structure in social environments have more comprehensive information and are much easier to manipulate. On the basis of discussion earlier, different from [25], our work takes advantages of both probing techniques and community features.

Based on the framework, they provided an influence maximization algorithm. We are also interested in the dynamic probing algorithm by taking a community as the unit instead of a node. As we stated, a community is a very natural structure in social networks. For a dynamic network, probing one node means updating the node itself and all the links related to the node. Nodes within the same community have more links and relationships that can affect each other. Probing the network node by node will result in a lot of duplicated operations and costs. In our solution, we look for the most active community and probe it. The cost is reduced by probing the community as a unit.

On the other hand, taking the community into account brings more benefits for the influence maximization. Individuals within the same community have more frequent contacts and thus are more likely to be influenced by each other. In contrast, individuals from different communities have much less contact with each other and thus are less likely to influence each other. This community property suggests that it might be a good approximation to choose influential nodes within communities instead of the whole network. Obviously, it will be more efficient to choose influential nodes within communities. Based on the previous discussion, we employed dynamic programming to take full advantage of the community and probing techniques. Reference [25] only provided the node probing method while excluded the community probing and other techniques related to communities. Reference [25] also does not take the efficiency of the influence maximization into account.

Li *et al*. [26] addressed the problem of finding densely connected subgraphs that satisfy the query conditions considering the influence of a community in a network. However, their method is based on the concept of *k*-core, which is not an influence diffusion–expectation model but network structure model. This kind of model could not provide influence expectation measures, which actually

Trans. Emerging Tel. Tech. 2017;**28**:e3054. © 2016 John Wiley & Sons, Ltd.
DOI: 10.1002/ett

do not completely follow the information diffusion process. This has been proved and verified in many other experiment-based literatures [27] [8] to some extent. Wang *et al*. [9] tried to reduce computation cost by dividing a network into many communities. The greedy algorithm was first run in each community, and the increments of the expected influence for each community were calculated. A dynamic programming algorithm was first proposed to select the optimal community. Then, the most influential nodes from each community were compared and selected. This process runs iteratively until the top-k influential nodes were found. Different from our work, timeliness had not been considered in their model. Besides, their partition method just divides a network into disjointed communities for the purpose of reducing computation cost without incorporating any other advantage of a network. Additionally, in real applications, communities in a social network naturally overlap because people have multiple roles in different situations for most social networks.

To the best of our knowledge, none of the existing approaches consider merging the dynamic network probing problem from the community scope with the community-based divide-and-conquer techniques together to solve the influence maximization problem.

3. PRELIMINARIES AND PROBLEM DEFINITION

We first introduce the probing technique for a dynamic network. The motivation behind it is that it is almost unrealistic to observe the whole dynamic network at any moment, but it is much easier to monitor the changes of some parts of the network. Initially, the entire network can be represented as $G = (V, E, W)$, where V is the node set, $E = V \times V$ is the edge set, and W is the weight set for edges representing the influence probability among users for a given diffusion process. Let $\mathbb{C}$ be the set of initially detected communities, C be one community that belongs to set $\mathbb{C}$, and C^i be the ith community.

Because the dynamism of networks results in the change of links in each community, we only consider the node set for a community. Assume there are n_C communities in total and then the union of all C^i from C^1 to C^{n_C} constitutes the whole network. For discrete time sequences $t = \{0, 1, \ldots, n\}$, let G_t be the network topology for the corresponding time stamp. Then G_0 is the actual observed one, and the network topology can only be observed by probing in the rest of time series. Assume budget b equal the number of communities that can be probed in one time stamp. If $b = n_C$, then the whole network will be updated in this time stamp. Otherwise, only b communities will be updated in the network to approximate the actual network topology.

To make the illustration more clear, we will briefly review the classical IC Model for the influence maximization [6]. The diffusion process under the *IC* model works in discrete time. Initially, all the seeds in set S are activated, while all the other nodes are inactive. In the next time step, any active node u in the previous step is given a single chance to activate any of its inactive neighbours with an independent probability $w(u, v) \in W$. Once a node is activated, it does not change its status anymore. The influence process continues until the number of activated nodes stops increasing.

Let $\delta(S)$ be an influence function representing the final number of activated nodes in a network when the initial activated node set is S. The objective is to find the most influential seed set S with size up to k such that they can maximize $\delta(S)$ in Equation (1).

$$S^* = \arg \max_{S \subseteq V, |S| \leqslant k} \delta(S) \tag{1}$$

Because the *LT* model has been proved that could be unified with proper parameter initialization to *IC* [28], we introduce *IC* and all our proposed algorithms that are equally applicable to *LT*.

Preliminary problem: Given the topology of a dynamic network G at time $t = 0$ and the changes that can be observed by probing, supposing b is the budget allowed to probe communities, the problem is how to accurately approximate the 'real' network topology at time t via selectively observing b communities and updating the remaining unselected communities based on the historical topology information.

Once the updated network topology is obtained, the next target is to maximize the influence in the network.

Ultimate problem: Given a dynamic social network G and the probing budget b, while probing pieces of the network at each time stamp to approximate the change of the whole network, we aim to find a set S with size k such that $\delta(S)$ is maximized under the specified classical (e.g. *IC* or *LT*) influence diffusion model.

4. MODEL AND ALGORITHM

Based on the problem defined in Section 3, the input of our problem is a dynamic network G. The expected outcome is to approximate the real network change by probing the change of b communities in the network and then find up to k seeds, which can maximize the influence considering the dynamics of the social network. Later, we first present an overview of our partition method based on [29], which prepares communities for our following probing algorithm. We propose a community partition algorithm in order to further improve the efficiency and effectiveness of our solution. The influence diffusion among nodes has been taken into consideration in the community partition process to make it more applicable to our defined model. The partition algorithm includes three steps. First, we assign unique community labels from 1 to $N = |V|$ to each node. Second, we compute the influenced neighbour set for each node based on the *IC* model. Third, labels are

Table I. Notation summary.

Notation	Description
$\mathcal{G} = (V, E, W)$	A weighted directed graph
V	The node set
E	The edge set
W	The weight set for edges
k	# of influential nodes to be mined
S	The set of initial activated nodes
S^*	The set of final activated nodes
$\delta(\cdot)$	The influence maximization function
C^i	The ith community
n_C	The number of communities
b	The budget to probe communities
$w(u, v)$	The probability on edge (u, v)
$\widehat{E}(S)$	The influence function of S
ϵ	The parameter of probing quality
$\varpi(C)$	The bound of probing quality
τ_C	The percentage parameter for probing

propagated through the network. Table I summarizes the notations used in this paper.

4.1. Probing dynamic networks

A simple strategy for community-based network update is to randomly choose communities with equal probability and then probe the changes within each community. In this case, communities that have not been changed may be selected, and communities that have been changed may not be selected. We may lose accuracy from inaccurate approximation. A better solution is to accurately find communities that are expected to bring the biggest change in the network, so that probing the updating of the selected communities can be used to closely estimate the updating of the network. Assume $\widehat{E}(S)$ evaluates the influence spread from seed set S in the observed network G after probing the community C. Let S_m and S'_m be the maximized influence seed set obtained before and after community C has been probed, respectively. Because S'_m depends on C, the function could also be written as a function of C, *i.e.*, $S_m(C)'$. The larger $(\widehat{E}(S_m) - \widehat{E}(S'_m))$ is the more meaningful for probing a specific C. We expect that the occurrence probability of such difference is no more than ϵ. Accordingly, let $\varpi(C)$ be the bound of the difference $(\widehat{E}(S_m) - \widehat{E}(S'_m))$. The objective of our probing algorithm is to optimize the approximation of global topology changes by only probing parts of communities. And for each community C, the maximum bound $\varpi(C)$ is satisfying

$$P[|\widehat{E}(S_m) - \widehat{E}(S'_m)| \geqslant \varpi(C)] \leqslant \epsilon \tag{2}$$

As we mentioned, computing the expected influence spread for a given seed set is intractable even in a relatively small network. Thus, we employ the Azuma–Hoeffding inequality, which provides the method of bounding differences. This method is commonly used in the analysis of randomized algorithms to estimate ϵ instead of directly calculating the influence spread.

A heuristic function $\Delta(C)$ based on the classical result of Chen [30] is developed:

$$\begin{aligned}\Delta_\Pi(C) =& 1 + d_{in}(C) - s_{in}(C) - s_{out}(C) \\ & - s_{out}(C)[d_{in}(C) - s_{in}(C)]\end{aligned} \tag{3}$$

where Π denotes the currently selected seed set, $d_{in}(C)$ is the total in-degree of nodes in community C $s_{in}(C)$ and $s_{out}(C)$ are the number of nodes that have already been chosen as seeds in Π among the predecessors and successors from community C, respectively. Based on this heuristic function, we could calculate the marginal improvement $\widehat{\Delta_\Pi}(C)$ of each community C in the observed network $\widehat{G}$. Note that, for a community C and a specific node u (which could be chosen or not chosen as a seed node), we consider C as chosen if and only if more than τ_C percentage of nodes in C have been chosen as seeds. Although the network is dynamic, the summation of in-degree of nodes in each community should be relatively stable; thus, the expectation is $E[d_{in}^{t+1}(C)|d_{in}^t(C)] = d_{in}^t(C)$.

The assumption that in-degree of nodes in each community is relatively stable is based on our observation and several previous related literatures. Zhuang *et al.* [25] adopted similar assumption for the stainability of a individual node. Our assumption is a relaxed version of the assumption in [25] , and we consider that the in-degree of nodes in a whole community is relatively stable, which could also reduce the effect of nodes' individual changes. In each time slot, for a specific node, the in-degree is the number of direct link point to that node. In social networks especially online social networks (*e.g.* Facebook and Twitter), most of the links are generated from friendships and interactive communications. Once a link between two nodes has been build, keeping the link has no extra cost. Furthermore, it is not common to see the situation that many links directed to one node are removed at the same time in reality. In some cases, relationships change among the nodes within the same community that may not affect the in-degree of the community. For example, the communication between A and B changes to A and C (assume user A, B and C are in the same community). Thus, even though the in-degree of a single node may change, the stability actually will be increased while rising the horizon from a node to a community. Previous work in [31], Kumar *et. al* analysed users' behaviour from the community level in [32], and they describe the pattern of users' behaviour in a social networks and [32] reported the stability of community structure in social networks. According to their research, the in-degree is one microcosmic aspect of a community, which should naturally follow the same properties. From empirical datasets [33] and observations obtained from [32] also demonstrates that networks keep relative stability in real social networks, which confirm our assumption again. The results show the ties among people who are more likely to share opinions (*e.g.* same race, gender or socioeconomic class) decay at a slower rate than ties among persons who are likely to have different opinions. For the link of influence, the ties are much more stable in

Trans. Emerging Tel. Tech. 2017;**28**:e3054.
DOI: 10.1002/ett

the social scenario. Therefore, we believe that the stability of the in-degree in a community of a social network is a reasonable assumption.

Suppose the latest update of the network is at time t_δ, and $|d_{in}^{t+1} - d_{in}^{t}| < t_\delta$. Applying the Azuma–Hoeffding inequality for martingale, we have

$$P\left(d_{in}^{t+t_\delta}(C) - d_{in}^{t}(C) \geqslant z\right) \leqslant e^{\frac{-z^2}{2t_\delta}} \tag{4}$$

Because d_{in} is sub-martingale, symmetrically

$$P\left(d_{in}^{t+t_\delta}(C) - d_{in}^{t}(C) \leqslant -z\right) \leqslant e^{\frac{-z^2}{2t_\delta}} \tag{5}$$

With Equations (4) and (5), by applying the union bound, we have

$$P\left(|d_{in}^{t+t_\delta}(C) - d_{in}^{t}(C)| \leqslant -z\right) \leqslant e^{\frac{-z^2}{2t_\delta}} \tag{6}$$

Let the probability in Equation (6) to be bounded by ϵ. We have $e^{\frac{-z^2}{2t_\delta}} = \epsilon$, then $z = \sqrt{-2t_\delta \ln \epsilon}$. If we probe $C \in S_m$ and find that $\widehat{d_{in}}(C)$ is still higher than $\arg\max_{C' \notin S_m} d_{in}(C')$, then the performance gap will be 0; otherwise, C will not be included by $S'_m(C)$, and $\dot{C} = \arg\max_{C' \notin S_m} d_{in}(C')$ will be included. Thus, the difference before and after probing C would be $\min\{0, \widehat{d_{in}}(\dot{C}) - d_{in}(C)\}$ if $C \in S_m$, and $\min\{0, d_{in}(C) - \widehat{d_{in}}(\ddot{C})\}$ where $\ddot{C} = \arg\min_{C'' \in S_m} \widehat{d_{in}}(C'')$ if $C \notin S_m$.

Thus, the value of function $\varpi(C)$ is as follows:

$$\begin{cases} \min\{0, \widehat{d_{in}}(C) - z - \min_{\dot{C} \in S} d_{in}(\dot{C})\} & \text{if } C \notin S_m \\ \min\{0, \max_{\ddot{C} \notin S} \widehat{d_{in}}(\ddot{C}) - d_{in}(C) + z\} & \text{if } C \in S_m \end{cases} \tag{7}$$

Based on the estimation of $\varpi(C)$, Algorithm 1 is proposed to update a dynamic network by probing the change of several important communities within the network. In Algorithm 1, the initialization has been shown in Line 1. The probing process is an iteratively repeated process as shown from Lines 2 to 21, where Lines 4 to 15 show one iteration. Lines 17 to 21 update and print the result set S_m^t of time t, which will be discussed in more detail in the next section. The overall complexity of Algorithm 1 is $t * (O(bn_C) + O(IM))$ where $O(IM)$ is the cost of influence maximization.

Algorithm 1: Probing algorithm

Input: network G, budget b, time t, threshold ϵ
Output: Seed set S_m^t at $t \in T$

1 Initialize $\widehat{G} = G$; $\forall t_\delta = 0$;
2 **for** $t \in T$ **do**
3 $\forall C \in V, t_\delta = t_\delta + 1$;
4 **for** $i = 1$ *to* b **do**
5 $S_m = k$ nodes with maximum $\hat{d}_{in}(C)$;
6 $\hat{d}_{max} = \max_{C \notin S_m} \hat{d}_{in}(C)$;
7 $\hat{d}_{min} = \min_{\dot{C} \in S_m} \hat{d}_{in}(\dot{C})$;
8 **for** $C \in V$ **do**
9 $z_C = \sqrt{-2t_\delta \ln \epsilon}$
10 **if** $C \in S_m$ **then**
11 $\min\{0, \hat{d}_{in}(C) - z - \min_{\dot{C} \in S} d_{in}(\dot{C})\}$
 end
12 **else**
13 $\min\{0, \max_{\ddot{C} \notin S \hat{d}_{in}(\ddot{C})} - d_{in}(C) + z\}$
 end
 end
14 $C^* = \arg\max_{C \in V}\{\varpi(C)\}$
15 Probe C^* in G_t and update $\hat{G}$
 end
16 // Community-based algorithm in the next section
17 **for** $i = 1$ *to* k **do**
18 $C^* = \arg\max_{C \in V \S_m^t} \hat{\Delta_\Pi}(C)$
19 $S_m^t = S_m^t \cup C \cap S_m^t$
20 Update $\hat{\Delta}_\Pi(C)$
21 **return** S_m^t
 end
end

4.2. Influence maximization in communities

We first briefly give the algorithm of community detection. Our community detection algorithm is based on the classical algorithm proposed in [34]. The main idea is assigning a unique community label from 1 to $|V|$, where $|V|$ is the number of nodes. Then the algorithm performs the label propagation among a node and its neighbours based on the *IC* model. At each iteration of the label propagation, the community detection algorithm assigns the community label for a node v based on the labels of its neighbours that are influenced by v. The algorithm stops until we go through the dynamic time slot.

One straightforward method that can be used to find the top-k influential nodes is to obtain the top-k influential nodes in each community, locally, and then find the global k nodes based on the local results. However, this approach results in a high computation cost, most of which is mainly caused by the repeated computation of local communities. For example, we have n_C communities, and if we find top-k nodes in each community by a naive algorithm, before we obtain the final result, we need to generate at least $n_C \times k$ candidates. But the final result only contains k nodes. That is, the calculation of the other $n_C \times k - k$ nodes is a waste if skipping any of them does not influence the final result. More local results also involve more operations in further filtrating.

To reduce the cost, we utilize the dynamic programming to find communities that could provide the global kth influential nodes. Wang *et. al.* [9] proposed an algorithm named *CGA*, which provides a solution to do the influence maximization based on communities. However, their algorithm cannot be directly employed in dynamic networks. Furthermore, the community detection algorithm used in [9] prevents overlap of communities in the network, which

is not practical in reality. Compared with *CGA*, our algorithm has the following improvements: (i) A more practical community detection algorithm, which allows community overlap, has been proposed. The detailed algorithm is ignored because of space limitation. The main idea is based on the algorithm proposed in [29], which divides a network according to the structure similarities. (ii) Allowance of overlap means that some nodes can participate in different communities. We build a max-heap to store all the intermediate results, which guarantees the worst adjustment time cost is limited within $O(n \log n)$. Based on how much enhancement of influence derived from each node, the algorithm ranks nodes with time. (iii) *CGA* utilizes the diffusion speed to measure the speed of influence. In practice, the speed of influence in a social network is based on the distance of connection among users, which should be measured by the number of minimum hops or the activated nodes on the connected path. Different from *CGA*, our probing algorithm takes the update of a whole network into account but ignores the calculation of the diffusion speed.

Dynamic programming gives us a way to obtain results through iterations by keeping intermediate results in a table. Let S^*_{i-1} be the set of influence nodes obtained in the $(i-1\text{th})$ step through the influence maximization process. If we find the ith influential node in community C^l, denoted the maximal increase as $\triangle\delta_l$, then we have

$$\triangle\delta_l = \arg\max \left\{ \delta_l \left(S^*_{i-1} \cap v_j \right), \delta_l \left(S^*_{i-1} \right) | v_j \in C^l \right\} \quad (8)$$

The notation v_j represents a new node added to the objective function, which belongs to community C^l, and $\Delta\delta_l$ is the local influence increment brought by v_j's joining. It is worth mentioning that the influence in Equation (8) is computed only in local community C^l rather than the whole network, which is pretty fast in practice. To evaluate which community is a better choice, the influence increment on one community is used as the measurement. The community that brings the largest increment will be the best candidate to find the ith influential node.

Let $S[l,i]$ denote the total influence of finding ith influential node in the first l communities, where $l \in [1, n_C]$, $i \in [1,k]$, $\delta[l,0] = 0$ and $\delta[0,i] = 0$.

$$\delta[l,i] = \arg\max\{\delta[l-1,i], \delta[n_C, i-1] + \triangle\delta_l\} \quad (9)$$

Equation (9) is a typical dynamic programming process. When we find the ith influential node, if the influence function $\delta(\cdot)$ has higher value in the first $l-1$ communities than in the l communities, we mine it from the first $l-1$ communities. Otherwise, we mine it from the previous l communities.

Algorithm 2 shows the process of finding the top k influential nodes in a network after probing the dynamic of our solution. Lines 1 and 2 show the initialization. In the loop (lines 3 to 17), according to the equation, Lines 3 to 14 show the process of maintaining a heap H and storing all the information and the influence incremental value of the tested nodes. After sorting the result in each community, Line 17 outputs the final result. The complexity of Algorithm 2 is $O(|L|k^2)$. Because k generally is much smaller than nodes number n, Algorithm 2 is almost a linear algorithm except Line 16. Our algorithm limits the calculation of influence maximization within each community, which could speed it up compared with global network computing.

5. EVALUATIONS

To evaluate the performance of our proposed approaches, we test our algorithms on synthetic data based on well-studied models as well as real social networks. To test the effects of network size, topology and dynamics on our proposed solution, we generate some synthetic dynamic networks. Two different models, small-world graph and Kronecker graph models, are applied to generate networks. Besides generated networks, real social networks are also used to verify the performance of our proposed solution.

Algorithm 2: Community-based dynamic algorithm

```
   Input: network G, size of result k
   Output: Top influential node set with size k
1  Initialize communities to L with n_l = |L|, and all
   δ[l,0] = 0;
2  δ[0,k] = 0, heap H = null, result R_{l,k} = null;
3  for i = 1 to k do
4      for l < n_l do
5          △δ_l =
           arg max{δ_l(S*_{i-1} ∩ v_j), δ_l(S*_{i-1})|v_j ∈ C^l}
6          δ[l,i] =
           arg max{δ[l-1,i], δ[n_C, i-1] + △δ_l};
7          if δ[l-1,i] ⩾ δ[L,i-1] + △δ_l then
8              R_{l,i} = R_{l-1,i};
9              Update the heap H for all tested nodes;
10             // Mine the i^th influential node from first
               l-1 communities;
           end
11         else
12             R_{l,i} = l;
13             Update the heap H for all tested nodes;
14             // Mine the i^th influential node from the
               mth community;
           end
       end
15     for i = 1 to k do
16         According to R_{l,k}, calculate the most
           influential nodes in each community by IM
           algorithm [37];
       end
   end
17 return k influential nodes
```


Trans. Emerging Tel. Tech. 2017;**28**:e3054. © 2016 John Wiley & Sons, Ltd.
DOI: 10.1002/ett

Table II. Details of synthetic data.

Network model	Nodes	Edges	Communities #
Syn-SmallWord(S)	2000	13657	15
Syn-SmallWord(L)	50000	468391	36
Syn-Kronecker(S)	2000	21062	17
Syn-Kronecker(L)	50000	723276	41

5.1. Data and observations

Data from synthetic generated social networks

Small-world graphs: the small-world network model is a very classical model following the small-world features according to 'small-world' [35]. This model is referred as *Syn-SmallWord*.

Kronecker graphs: this generative model proposed in [36] generates a network in a natural way. The networks grow from five initial nodes, and then Kronecker idea is repeatedly apply to expand the network. This model is referred as *Syn-Kronecker*.

Based on the initial networks generated from the previous models, we dynamically change each network based on the idea proposed in [37]. Because we have multiple synthetic networks in the experiments, the average summary of 10 networks' statistic features has been used instead. As shown in Table II, we generate two scales (small (S) and large (L)) of networks for both models with time stamp length of 50 and 100, respectively.

Data from real social networks

Epinions is a Who-trust-whom network, where nodes are users using their services and an edge from node u to another node v means u has influence on v (u is trusted by v). This network includes 75 879 nodes and 508 837 edges.

Slashdot is a technology-related news network, which features user-submitted and editor-evaluated contents. This network includes 77 360 nodes and 905 468 edges.

Both *Epinions* and *Slashdot* are obtained from Stanford Network Analysis Platform.[‡]

Twitter is one of the most notable micro-blogging service provider.[§] Twitter users can publish tweets (with a limited length of 140 characters). We use the dataset obtained from [38]. The subnetwork includes 112 044 nodes (users of Twitter), 468 238 edges (following relationships) and 2 409 768 tweets posted by the selected users.

Inventor is a network of inventors (extracted from USPTO[¶]) obtained from [39]. The edges in this network represent the co-inventing relationship. The *Inventor* network consists of 2 445 351 nodes and 5 841 940 edges.

Our work aims at probing partial communities in dynamic networks to maximize the influence from a global scale. All the previous datasets are from real static networks. Based on those datasets, dynamic networks are derived from randomly select nodes and communities.

[‡]http://snap.stanford.edu/data/.

[§]http://www.twitter.com.

[¶]http://www.uspto.gov/.

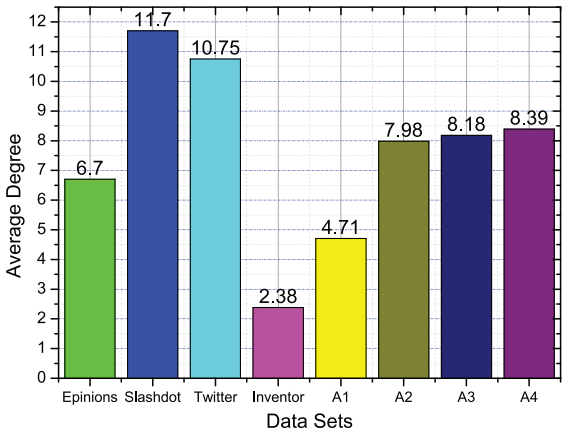

Figure 2. Average degree of the real datasets.

Table III. Amazon datasets.

Dataset	Nodes	Edges	Diameter
Amazon0302 (A1)	262 111	1 234 877	29
Amazon0312 (A2)	400 727	3 200 440	18
Amazon0505 (A3)	410 236	3 356 824	21
Amazon0601 (A4)	403 394	3 387 388	21

Particularly, the Amazon co-purchase network is a network with real dynamic observations that had been used to verify the effectiveness and efficiency of our algorithm.

Amazon Dynamic Networks is based on the *Customers Who Bought This Item Also Bought* feature of the Amazon website. The four datasets are from March to May in 2003. There is a directed edge from product i to product j if i is frequently co-purchased with j [7].

Shown in Figure 2 is the average degree of all the eight real datasets we have used. Most research literatures assume that the probabilities or the weights on edges and the thresholds are given as the input. However, as pointed out in [11], learning those probabilities and thresholds has remained a non-trivial problem. Therefore, we use the learning algorithms proposed in [40] to achieve the balance between complexity and practicability to the raw input data for all the synthetic generated and real networks. For the Amazon datasets, as shown in Table III, because they are a series of snapshots from the network, we generate the real influence spreading trend by comparing our model with the real learning algorithm proposed in [41][42], which initially treats the data as the user log then solves the influence maximization.

The probability is learned from the networks in the previous position. That is, the probability of the second network comes from the first one, and the probability of the last network is generated from the first three network snapshots through the linear prediction.

5.2. Algorithm evaluation

In both synthetic generated and real networks, we evaluate algorithms proposed in Section 4 based on the classical

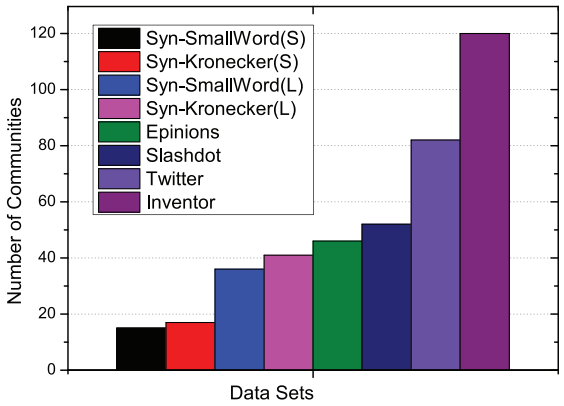

Figure 3. Number of communities in each network.

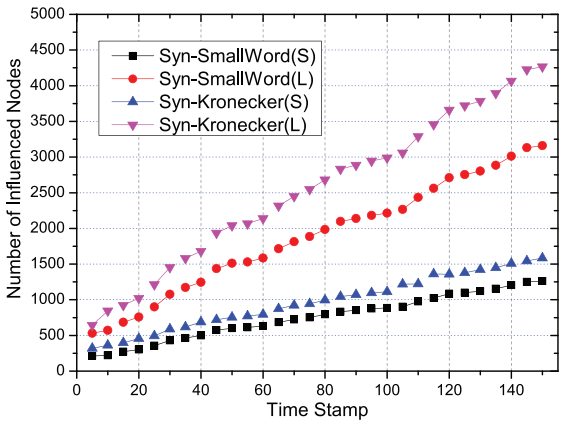

Figure 4. Results of the probing algorithm with $b = 2$.

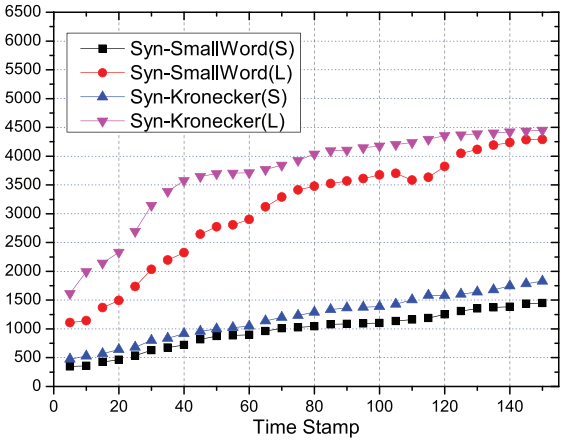

Figure 5. Results of the probing algorithm with $b = 10$.

approach introduced in [29]. Several optimizations are used to partition a network. The community detection is a pre-process for all the data. Communities are detected by initially assigning each node with a unique label and adopting the label step by step for most of its current neighbours. In this case, iteratively, the most densely connected groups will be grouped into a community. The result is shown in Figure 3, from which we can see that as the network size increases, the number of communities increases accordingly.

All the codes are implemented in Python 2.7 based on the latest version of Snap.py,[‖] and all experiments are performed on a PC running Windows 10 with Intel(R) Core(TM) i3-2120 CPU 3.30-GHz and 12-GB memory.

We first evaluate the probing algorithm and then test the overall efficiency. Comparisons are conducted with the following algorithms.

- Random nodes probing (*RandNodes*). Randomly choose $b*|\overline{C}|$ nodes, where $|\overline{C}|$ is the average community size in the network and then probe communities with uniform probability at each time stamp.
- Random community probing (*RandComm*). Randomly choose b communities and probe with uniform probability at each time stamp.
- Maximum gap probing (*MaxG*). This algorithm is proposed in [25]. Choose $b * |\overline{C}|$ nodes according to their algorithm.
- Optimized communities probing (*OptComm*). This is our algorithm introduced in Section 4. Probe b communities per time stamp.
- Global updating the whole network (*OptWhole*). This is the optimal benchmark of updating a whole network in each time stamp.

We compare all the algorithms from the aspects of effectiveness and efficiency. In order to be more reasonable, we calculate the influence expectation based on the uniform algorithm proposed in [18]. Later in this section, we will show the advantages of our integral solution.

[‖]Python interface for SNAP [43].

We evaluate the performance of our probing algorithm on the synthetic networks by time stamp. Figures 4 and 5 show the results of four different synthetic networks with different probing budget b. We can see that as b increases, the beginning increments of influenced nodes are sharper, and then they go to a gentle slope. With more time stamps consumed, the algorithm could probe more accurate structure and property of a network, and then more nodes are influenced through the algorithm.

We divide our data into two groups: normal size group (Epinions and Slashdot) and large size group (Twitter and Inventor). With budget $b = 5$, Figures 6 and 7 show the number of influenced nodes and the running time, respectively.

Considering the number of the influenced nodes in five different algorithms, our *OptComm* is as good as *MaxG* and approaches to the performance of the optimal algorithm *OptWhole*. From the result, *MaxG* is a little bit better than our *OptComm*. That is because their algorithm considers a node as a unit. It finds the most active node and probe it. While, our algorithm considers a community as a unit to reduce computational cost, which may miss the most active node. *OptComm* takes the advantage of considering

Trans. Emerging Tel. Tech. 2017;**28**:e3054. © 2016 John Wiley & Sons, Ltd.
DOI: 10.1002/ett

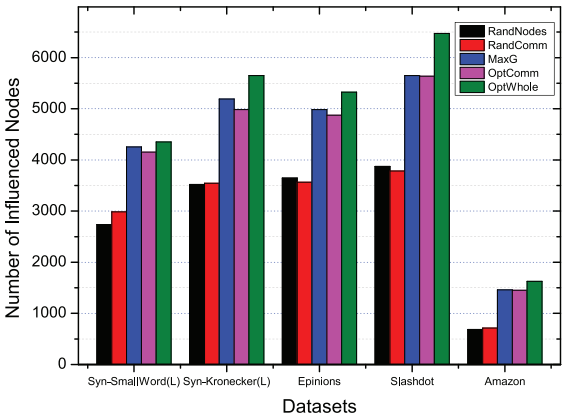

Figure 6. Number of influenced nodes in the probing algorithm.

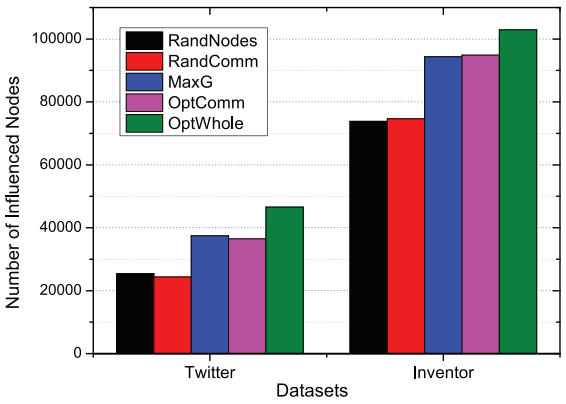

Figure 8. Number of influenced nodes in the probing algorithm.

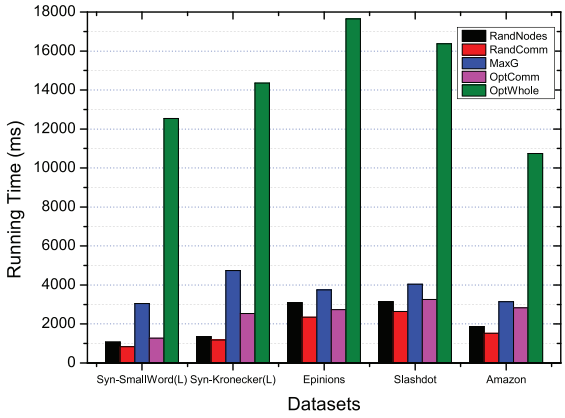

Figure 7. Running time of the probing algorithm.

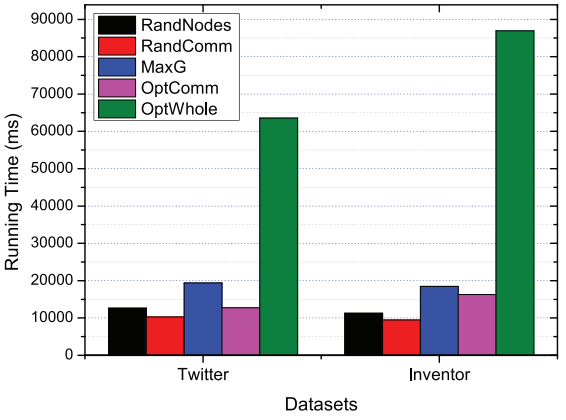

Figure 9. Running time of the probing algorithm.

community as a unit, which partly save computational cost of probing nodes within the same community. This strategy could calculate a more accurate influence within the same community especially when the community structure is sparse. This is the situation of *MaxG* and *OptComm* in the Inventor network. Inventor has 120 communities compare with 82 communities of Twitter. At the same time, as shown in Figure 2, Inventor has the smallest average degree 2.38 compared with 10.75 of Twitter.

The average degree of Twitter (10.75) is much higher than Inventor (2.38). According to the results in Figures 8 and 9, the running time of OptComm is much better than MaxG in Twitter compared with Inventor. We can easily find out that the difference of the two networks in the number of influenced nodes in Twitter is much higher than Inventor. This is resulted of different sparseness of different networks. Therefore, with the same probing budget, for a sparse network, algorithm *OptComm* could update the network more accurate and achieve better performance.

The advantage of our *OptComm* is when we take a community as probing unit, although the community has $|C|$ nodes, the real number of nodes needed to be updated is fewer than $|C|$ because most nodes in a community have strong connections with each other and most connections may have already been updated through previous operations. Even more distinct, as shown in Figure 7, benefitting from our community-based probing algorithm, our *OptComm* is much faster than *MaxG*, and even close to the random algorithms. The two random selection approaches are approximately the same, but the community-based algorithm has a better performance, which verifies that influence diffusion is community orientated. Both observations state that Inventor is a very sparse network, which further confirm that the advantage of algorithm *OptComm*. *OptComm* is only a little bit better than *MaxG* as shown in Figure 8.

We apply our algorithm to larger size networks. Results are shown in Figures 8 and 9. Contrasting with Figures 6 and 7, the larger the data size, the advantage of the community-based probing algorithm is more obvious. Especially, Figure 9 shows that the running time of our algorithm is only 2827 ms, which is 26.3% of the *OptWhole* and better than *MaxG*'s running time. Considering the overall performance of our solution, the running time as presented in Figure 7, *OptComm* is much better than *MaxG*, and this result further verified our model and algorithm.

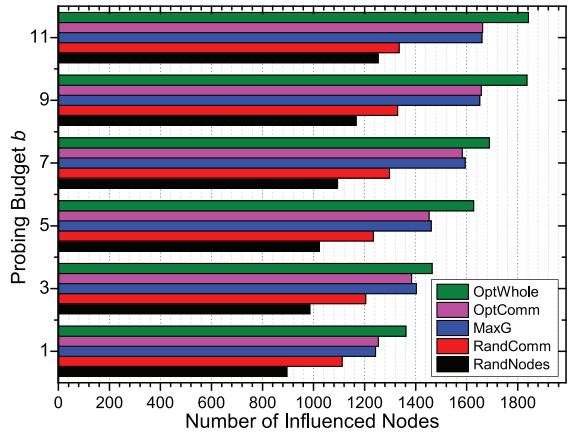

Figure 10. Effect of budget *b*.

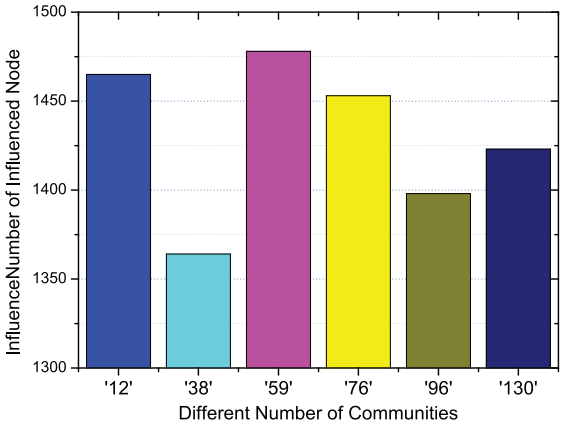

Figure 11. Number of influenced nodes with different numbers of communities.

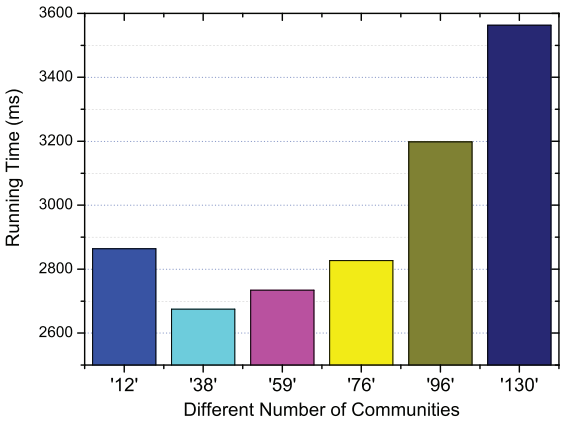

Figure 12. Running time comparison with different numbers of communities.

We separate the running time into two parts in order to demonstrate the advantage of our algorithm more clear. In the first step, our probing algorithm takes advantage of community probing, which is comparable with node-based probing on both accuracy and efficiency. In the second step, based on the community-based probing algorithm, we maximize the influence from local to global. The dynamic programming technique allows us to take full advantage of efficiency and achieve our contribution. Figure 13 shows the total running time, from which we can see that our *OptComm* is significantly better than *MaxG*.

The budget b is the parameter that balance the degree of dynamic approximation and the computational cost. Larger budget b gives more accurate approximation results. If budget b is equal to the total number of communities in the network, the algorithm could obtain the real update to the dynamics of the whole network.

As shown in Figure 10, more nodes are influenced as b increases. But as we increase the budget to a relatively large level such as 9 to 11, the more communities we probe, the marginal increment of influenced nodes is decreasing. This is because when the budget is large enough, more inactive communities will be tested, which could not contribute much to the final result because of their inactivity. On the other hand, the diminishing of marginal utility once again verifies that it is not necessary to update the whole network with massive cost. At the same time, larger budget of probing results in higher computational cost. Our probing solution is targeted at saving the computational cost. It is not very meaningful to use an enormous budget. In the experiment section, we test the performance of different budget b in different datasets.

From the real-data experiment result, budget b from 3% to 5% of the total communities could achieve the best balance between efficiency and computation overhead. Therefore, in our evaluation section, budget $b = 5$ is used in most of experiments, which is a typical value we have tested. Other budget also follows the similar trend.

According to our analysis and experiment results, the most important factor of the probing cost is the budget b. Different topology will affect the community structure but has no effect the probing cost significantly. To proof this, we changed the setting of the community detection algorithm to obtain different community detection results. Besides our default community detection result of 76 communities, we had another five different scale of community detection results as 12, 38, 59, 96 and 130, respectively. As shown in Figure 11, as well as the increase number of communities, the number of influenced nodes is changing fluctuated.

In order to further investigate how the community was affecting our algorithm cost, we also compared the cost of different community scales as shown in Figure 12. The real experiment result shows that when the number of communities is very small, which implies that each community is very large, the cost of probing is increased while the benefit of community probing is decreased, but the trend is kept in a relative smooth way. In contrast, when the number of communities is increasing, the cost is also increasing because very small communities are generated in the network. Another reason for the increase of cost is that when we have more communities, we need probing and compute

Trans. Emerging Tel. Tech. 2017;**28**:e3054. © 2016 John Wiley & Sons, Ltd.
DOI: 10.1002/ett

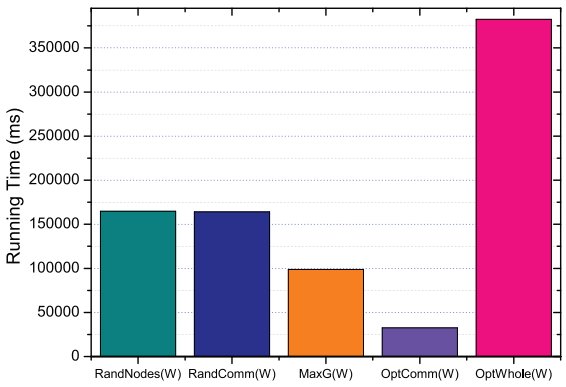

Figure 13. Overall running time comparison.

more communities during both steps in our solution. From the result, we can determine that the most important factor on probing cost is how many nodes we need to probe in total but not the number of communities. As we have tested, with the same number of nodes, investigating nodes by community can boost the efficiency of the whole algorithm even though, from both Figures 11 and 12, the change of experimental results is still kept in a relatively small range (for influenced nodes from 1300 to 1500 and for running time from 2500 to 3500 ms). This result states that the community structure does not have a strong effect on our algorithm. But other concerns such as probing techniques and dynamic programming algorithm do affect the final result.

In the end, we apply Algorithms 1 and 2 with probing budget $b = 5$ and $k = 50$ as an integral solution to evaluate the performance towards the Amazon data. As shown in Figure 13, our running time is around $1/3$ of *MaxG*'s and less than 10% of the *OptWhole*'s.

Overall, the proposed probing and dynamic programming algorithms as an integral solution show its outstanding performance regardless of random heuristic strategy or up-to-date techniques. In particular, the algorithms could be well employed to networks in a larger size, which indicates its high practicality and scalability.

6. CONCLUSION

In this work, we propose a practical dynamic probing framework to explore the real changes of networks. The probing framework takes the community as a unit and updates network topology by only probing b communities instead of searching the entire network. Besides, we propose a divide-and-conquer strategy to apply the dynamic programming technique to community-based influence maximization. The comprehensive experiment results show that our model can achieve comparable influence diffusion performance compared with the node-based probing algorithm, while having a much better computation cost, efficiency and more applicable to large-scale networks.

ACKNOWLEDGEMENTS

This work is partly supported by the National Science Foundation (NSF) under grant NO.CNS-1252292, NSF of China under grant 61373083, 61370084, 61502116, 61371185 and 61373027, NSF of Shandong Province under grant ZR2012FM023 and China Postdoctoral Science Foundation NO.2015M571231

REFERENCES

1. Fu Z, Sun X, Liu Q, Zhou L, Shu J. Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing. *IEICE Transactions on Communications* 2015; **E98-B**(1):190–200.
2. He Z, Cai Z, Wang X. Modeling propagation dynamics and developing optimized countermeasures for rumor spreading in online social networks. In *The 35th IEEE International Conference on Distributed Computing Systems (ICDCS)*, Columbus, Ohio, USA, 2015.
3. Xia Z, Wang X, Sun X, Wang B. Steganalysis of least significant bit matching using multi-order differences. *Security and Communication Networks* 2014; **7**(8):1283–1291.
4. Fu Z, Ren K, Shu J, Sun X, Huang F. Enabling personalized search over encrypted outsourced data with efficiency improvement. *IEEE Transactions on Parallel and Distributed Systems (TPDS)* 2015. DOI: 10.1109/TPDS.2015.2506573.
5. Peng X, Wang J, Zhang Z, Xiao Q, Li M, Pan Y. Re-alignment of the unmapped reads with base quality score. *BMC BIOINFORMATICS* 2015; **16**(Suppl 5):S8.
6. Kempe D, Kleinberg J, Tardos V. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, D.C., 2003; 137–146.
7. Leskovec J, Krause A, Guestrin C, Faloutsos C, VanBriesen C, Glance C. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2007; 420–429.
8. Rossi RA et al. Modeling dynamic behavior in large evolving graphs. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*, Rome, Italy, 2013.
9. Wang Y, Cong G, Song G, Xie K. Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, USA, 2010; 1039–1048.

10. Han M, Yan M, Cai Z, Li Y. An exploration of broader influence maximization in timeliness networks with opportunistic selection. *Journal of Network and Computer Applications* 2016; **63**:39–49.
11. Goyal A, Bonchi F, Lakshmanan LVS. Learning influence probabilities in social networks. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, New York, USA, 2010; 241–250.
12. Saito K, Kimura M, Ohara K, Motoda H. Learning asynchronous-time information diffusion models and its application to behavioral data analysis over social networks. *Journal of Computer Engineering and Informatics* 2013; **1**(2):30–57.
13. Han M, Yan M, Li J, Ji S, Li Y. Neighborhood-based uncertainty generation in social networks [J]. *Journal of Combinatorial Optimization (JOCO)* 2014; **28**(3): 561–576.
14. Tang J, Sun J, Wang C, Yang Z. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, 2009; 807–816.
15. Goyal A, Lu W, Lakshmanan LVS. Celf++: optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th International Conference Companion on World Wide Web*, Hyderabad, India, 2011; 47–48, 1963217.
16. Goyal A, Bonchi F, Lakshmanan LVS. Venkatasubramanian, approximation analysis of influence spread in social networks. *arXiv preprint arXiv* 2005: 1008.
17. Wang C, Chen W, Wang Y. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery* 2012; **25**(3):545–576.
18. Chen W, Wang C, Wang Y. 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, USA, 2010; 1029–1038.
19. Tang J, Wu S, Sun J. Confluence: conformity influence in large social networks. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2013; 347–355.
20. Li J, Cai Z, Yan M, Li Y. Using crowdsourced data in location-based social networks to explore influence maximization. In *The 35th Annual IEEE International Conference on Computer Communications (INFOCOM)*, San Francisco, CA, USA, 2016.
21. Anadiotis A, Christos G, Patrikakis CZ, Murat Tekalp A. Information-centric networking for multimedia social and peer-to-peer communications. *Transactions on Emerging Telecommunications Technologies* 2014; **25**: 383–C391.
22. Duan Z, Yan M, Cai Z, Wang X, Han M, Li Y. Truthful incentive mechanisms for social cost minimization in mobile crowdsourcing systems. *SENSORS-BASEL* 2016; **16**(4):481.
23. Rasool E, Nadia KH, Faramarz H, Mostafa SHS. A vehicular network for social services using data dissemination. *Transactions on Emerging Telecommunications Technologies* 2014. DOI: 10.1002/ett.2911.
24. Zhang L, Wang X, Lu J, Ren M, Duan Z, Cai Z. A novel contact prediction-based routing scheme for DTNs. *Transactions on Emerging Telecommunications Technologies* 2014. DOI: 10.1002/ett.2889.
25. Zhuang H, Sun Y, Tang J, Zhang J, Sun X. *Influence maximization in dynamic social networks*, 2013, 1313–1318.
26. Li R, et al. Influential community search in large networks. In *Proceedings of the VLDB Endowment*, Kohala Coast, Hawaii, USA, 2015; 509–520.
27. Cha M, Mislove A, Gummadi KP. *A measurement-driven analysis of information propagation in the flickr social network*, 2009.
28. Tang Y, Xiao X, Shi Y. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2014; 75–86.
29. Zhou Y, Cheng H, Yu JX. Graph clustering based on structural/attribute similarities. In *Proceedings of the VLDB Endowment*, Lyon, France, 2009; 718–729.
30. Chen W, Wang Y, Yang S. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2009; 199–208.
31. Kumar R, et al. On the bursty evolution of blogspace. *World Wide Web* 2005; **8**(2):159–178.
32. Kumar R, Novak J, Tomkins A. Structure and Evolution of Online Social Networks in Link Mining: Models, Algorithms, and Applications. In *Proceedings of the 12th International Conference on World Wide Web*, New York, NY, USA, 2003; 568–576.
33. Lizardo O, et al. Analysis of opinion evolution in a multicultural student social network. *Procedia Manufacturing* 2015; **3**:3974–3981.
34. Raghavan UN, Albert REK, Kumara S. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E* 2007; **76**(3), 036106.
35. Watts DJ, Strogatz SH. Collective dynamics of small-world networks. *Nature* 1998; **393**(6684):440–442.
36. Leskovec J, et al. Realistic, Mathematically Tractable Graph Generation and Evolution, using Kronecker

Trans. Emerging Tel. Tech. 2017;**28**:e3054. © 2016 John Wiley & Sons, Ltd.
DOI: 10.1002/ett

Multiplication. In *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Berlin, Heidelberg, 2005; 133–145.
37. Barab A Si AASO, Albert REK. *Emergence of Scaling in Random Networks. Science* 1999; **286**(5439): 509–512.
38. Hopcroft J, Lou T, Tang J. *Who will follow you back? Reciprocal relationship prediction*, 2011, 1137–1146.
39. Tang J, Wang B, Yang Y, Hu P, Zhao Y, Yan X, Gao B, Huang M, Xu P, Li W, et al. *Patentminer: topic-driven patent analysis and mining*, 2012, 1366–1374.
40. Saito K, Nakano R, Kimura M. *Prediction of information diffusion probabilities for independent cascade model*, 2008, 67–75.
41. Goyal A, Bonchi F, Lakshmanan LVS. data-based approach to social influence maximization, A. In *Proceedings VLDB Endow 5(1)*, Seattle, Washington, USA, 2011; 73–84.
42. Gu B, Sheng V, Wang Z, Ho D, Osman S, Li S. Incremental learning for v-Support Vector Regression. *Neural Networks* 2015; **67**(C):140–150.
43. Leskovec J, Krevl A. *SNAP (Stanford Network Analysis Platform)*, 2015. http://snap.stanford.edu/snappy/index.html.