Fast graph centrality computation via sampling: a case study of influence maximisation over OSNs

Rui Wang, Min Lv*, Zhiyong Wu, Yongkun Li and Yinlong Xu

School of Computer Science and Technology,
University of Science and Technology of China,
Hefei, Anhui, China
Email: rwang067@mail.ustc.edu.cn
Email: lvmin05@ustc.edu.cn
Email: wzylucky@mail.ustc.edu.cn
Email: ykli@ustc.edu.cn
Email: ylxu@ustc.edu.cn
*Corresponding author

Abstract: Graph centrality computation, e.g., asking for the most important vertices in a graph, may incur a high time cost with the increasing size of graphs. To address this challenge, this paper presents a sampling-based framework to speed up the computation of graph centrality. As a use case, we study the problem of influence maximisation, which asks for the k most influential nodes in a graph to trigger the largest influence spread, and present an IM-RWS algorithm. We experimentally compare IM-RWS with the state-of-the-art influence maximisation algorithm IMM and IM-RW, and the results show that our solution can bring a significant improvement in efficiency as well as a certain extent improvement in empirical accuracy. In particular, our algorithm can solve the influence maximisation problem in graphs containing millions of nodes within tens of seconds with an even better performance result in terms of influence spread.

Keywords: random walk; sampling; graph centrality; online social networks; influence maximisation.

Reference to this paper should be made as follows: Wang, R., Lv, M., Wu, Z., Li, Y. and Xu, Y. (xxxx) 'Fast graph centrality computation via sampling: a case study of influence maximisation over OSNs', *Int. J. High Performance Computing and Networking*, Vol. X, No. Y, pp.xxx–xxx.

Biographical notes: Rui Wang is currently a Master student at University of Science and Technology of China. She received her Bachelors in Computer Science from Donghua University in 2016. Her research interests include graph analysis, graph storage and file system.

Min Lv received her Bachelors and a Masters in Applied Mathematics from Anhui University in 1999 and 2002, respectively and PhD in Applied Mathematics from University of Science and Technology of China in 2005. Then she has been a Post-Doctor in the School of Computer Science and Technology at University of Science and Technology of China for two years. She is currently a Lecturer in the School of Computer Science and Technology, University of Science and Technology of China. Her research interests are social networks, security and privacy, and combinatorial mathematics.

Zhiyong Wu received her Bachelors in Computer Science from Anhui University of Technology in 2015 and is currently working toward his Masters at the School of Computer Science and Technology, University of Science and Technology of China, Hefei, China. His research mainly focuses on graph mining, graph analysis and graph storage.

Yongkun Li is currently an Associate Professor in School of Computer Science and Technology, University of Science and Technology of China. He received his BEng in Computer Science from University of Science and Technology of China in 2008, and PhD in Computer Science and Engineering from The Chinese University of Hong Kong in 2012. After that, he worked as a Post-Doctoral Fellow in Institute of Network Coding at The Chinese University of Hong Kong. His research mainly focuses on performance evaluation and architectural design of networking and storage systems.

Yinlong Xu received his BS in Mathematics from Peking University in 1983, and the MS and PhD in Computer Science from University of Science and Technology of China (USTC) in 1989 and 2004, respectively. He is currently a Professor at the School of Computer Science and Technology at USTC, and is leading a research group in doing some networking and high performance computing research. His research interests include network coding, storage systems,

combinatorial optimisation, design and analysis of parallel algorithms, parallel programming tools, etc. He received the excellent PhD advisor award of Chinese Academy of Sciences in 2006.

This paper is a revised and expanded version of a paper entitled 'Fast graph centrality computation via sampling: a case study of influence maximization over OSNs' presented at 4th Conference on CCF BigData, Lanzhou, China, 11–13 October 2016.

1 Introduction

The graph centrality which asks for the most important vertices in a graph has many important applications (Gao et al., 2014; Schlegel and Wong, 2015; Weng et al., 2016). For example, it can be used to identify the most influential users in a social network, or the key transport hubs in an urban network and so on. The commonly used centralities include degree centrality referring to the number of connected edges of each node, closeness centrality referring to the average length of the shortest paths between a node and all other nodes in the graph, betweenness centrality referring to the times of a node being a bridge along the shortest path between two other nodes, and so on. Due to the importance of the graph centrality, lots of algorithms for improving the computation have been proposed in recent years (Brandes, 2001; Newman, 2005; Zhao et al., 2014).

One wide application of graph centrality is influence maximisation (IM), which was first formulated by Domingos and Richardson as an algorithmic problem (Domingos and Richardson, 2001; Richardson and Domingos, 2002). The definition of an IM problem is as follows: given a graph $G(V,E)$ and a positive integer k, select the set S of k nodes from G which can trigger the largest influence spread in the network under a given diffusion model. A typical application of IM is in viral marketing, which takes use of the 'word-of-mouth' effect. That is, a company may first choose an initial set of influential users and give them a discount or free samples, then rely on these initial users to attract their friends to further purchase, and finally a large amount of people may buy the product due to the influence spread through out the whole network.

To solve the IM problem, many approaches are based on some heuristics, e.g., directly choosing the k nodes with the highest degree or the k nodes with the highest closeness centrality (Brandes, 2001; Newman, 2005; Zhao et al., 2014). Even though these heuristic algorithms are usually time-efficient, they can not give any guarantee on the computation accuracy.

Kempe et al. (2003) proposed a greedy algorithm with an approximation ratio of $1 - 1/e \approx 63\%$, but with high computational complexity. Later on, some researchers further improved the greedy algorithm to reduce the time complexity, like CELF and CELF++ (Chen et al., 2010; Goya et al., 2011; Leskovec et al., 2007). However, the time cost of these algorithms is still very high due to the use of the greedy framework.

To overcome the high time complexity of greedy approaches, some algorithms based on community are proposed (Cao et al., 2010; Galstyan et al., 2009; Wang et al., 2010). The idea is to first divide the graph into several communities by using a community partition algorithm, and then transform the influence maximisation problem to a dynamic resource allocation problem among different communities. This kind of algorithms show a big improvement in time efficiency compared with greedy algorithm.

Though the algorithms based on community partition greatly reduce the time complexity of greedy approaches, they are still time-consuming to deal with a huge graph. Recently, Borgs et al. (2014) presented a near linear time influence maximisation algorithm. It first randomly samples some nodes to generate a super-graph, in which each edge corresponding to a set of nodes that could be influenced by a randomly selected node, as its influence estimation. Then it uses the greedy algorithm to select the top-k nodes. Inspired by this work, Tang et al. (2014, 2015) further presented two algorithms, TIM and IMM. In particular, IMM is the state-of-the-art algorithm and it can reach $O((k + l)(n + m) \log n/\epsilon^l)$ expected running time and return a $(1 - 1/e - \epsilon)$ approximate solution with probability of $1 - 1/n^l$.

Considering that existing approaches to solve IM are all performed over the whole graph, so intuitively, their time costs should increase rapidly along with the increase of graph size. Thus, for large graphs containing tens of millions of nodes and billions of edges, even IMM is still time-consuming due to the need of traverse over the whole graph.

To address the efficiency issue of IM in large graphs, in this paper, we propose a new computing approach called IM-RWS by using graph sampling. The main idea is to first get a set of subgraphs by sampling, then compute the influence values of nodes in each subgraph, and finally aggregate the influence values computed from all subgraphs to approximate the influence in the original graph. With this sampling framework, the time complexity can be reduced to $O(nT)$, where n is the number of sampled nodes in each subgraph and T is the number of subgraphs or the times of sampling. In particular, the time complexity of IM-RWS does not depend on the size of the original graph, but the number of sampled nodes and the times of sampling, which makes IM-RWS time-efficient even for large-size graphs.

To evaluate the time efficiency and accuracy of IMRWS, we conduct extensive experiments to compare IMRWS with the state-of-the-art algorithms on several

realworld OSNs. The results show that IM-RWS brings a significant improvement of time efficiency as well as a certain extent of improvement in accuracy. In summary, this paper makes the following main contributions.

- We introduce a new method to speed up the computation of graph centrality for large-scale graphs with sampling. In particular, our method first sample multiple relatively small subgraphs from the original large graph, then compute the centrality over the small subgraphs, and finally aggregate the computed results to estimate the original graph centrality.
- As a special study case of our method, we present an algorithm IM-RWS for solving the influence maximisation problem. In this algorithm, we use random walk to sample a certain number of subgraphs and adopt the idea of IM-RW algorithm to calculate the influence values of nodes in each subgraph.
- We conduct experiments with real-world datasets to validate the effectiveness and efficiency of our approach. Results show that our approach can significantly reduce the time cost and meanwhile keeps the same or even higher accuracy.

Organisation. The rest of this paper is organised as follows. Section 2 describes preliminaries of the influence maximisation problem, and introduces the background on random walk. Section 3 first proposes the sampling-based framework and then presents our algorithm IMRWS for solving the influence maximisation problem. We show the experiment results in Section 4. Section 5 overviews the related works and Section 6 concludes the paper.

2 Preliminaries

In this section, we first introduce the diffusion model and then give the formal definition of influence maximisation problem. Finally, we overview random walk to facilitate the discussions of our algorithm in subsequent sections.

2.1 Diffusion models

Diffusion models define how the influence spreads through a social network. Usually, in different scenarios, the ways of influence spreading are also different. Thus, there are many diffusion models, such as *linear threshold cascade* (*LT*) model (Granovetter, 1978; Kempe et al., 2003), CTIC model (Saito et al., 2009), weighted cascade model (Kempe et al., 2003) and so on (Even-Dar and Shapira, 2007; Mahajan et al., 1991). The most commonly used model is *independent cascade* (*IC*) model (Kempe et al., 2003; Watts, 2002). The influence spread process under the IC model can be described as follows. The influence starts from an initial set of nodes called seeds. When a node u is influenced at timestamp t, then u will influence its out-neighbours with probability p at timestamp $t + 1$. The influence spreads until there is no more nodes can be influenced.

2.2 Influence maximisation problem

Definition: The *influence maximisation* problem (Domingos and Richardson, 2001; Richardson and Domingos, 2002) is defined as follows. Given a social network denoted as $G(V,E)$, a small positive integer k and a diffusion model, the goal is to find a node set with size k from V so as to influence the largest number of nodes, which can be mathematically formulated as follows:

$$A^* = \operatorname{argmax}\{\sigma(A) \mid |A| = k, A \subseteq V\}.$$

Here, $\sigma(A)$ denotes the number of nodes being influenced by node set A.

To evaluate the performance of an algorithm which solves the above influence maximisation problem, we consider the following two aspects:

Accuracy: It is characterised by the distance between the selected set of k seeds obtained from an influence maximisation algorithm and the optimal solution. However, since it is impossible to directly compute the optimal solution, especially for a large graph, instead, we measure the number of eventually influenced nodes by the selected k nodes. The rationale is that the more number of nodes that can be influenced by the k nodes selected by an algorithm, then the higher accuracy the algorithm achieves.

Efficiency: It is characterised by the time required to run an influence maximisation algorithm. Note that this measure is also important, because if the time cost of an algorithm is too high, then the algorithm may not be able to return a feasible solution in a reasonable time for large social networks.

2.3 Random walk

The process of simple random walk on social networks can be described as follows. It first randomly selects a node from the social network as the start point, then randomly selects a candidate node from the neighbours of the current node and walks to it. The above steps repeat until the termination condition is satisfied. During the process of random walk, there may be no candidate node to walk to in a directed graph. For instance, when the out-degree of the current node is 0, the walk can not continue. In order to overcome this drawback, at each step, we allow the walk to jump back to the start node and restart the walk with probability p, and with the rest probability $1 - p$, we continue the current walk.

There are lots of applications for random walk on social networks, such as PageRank (Page et al., 1999), similarity (Jeh and Widom, 2002) and so on (Li et al., 2014; Zhao et al., 2015; Zhong and Shen, 2006). Here, we just introduce two applications of random walk which are related to our work:

1 Random walk sampling (RW). It is an exploratory sampling method, which selects the nodes that are visited by the random walk process as sample nodes

2 Influence maximisation with random walk, e.g., IM-RW algorithm in Zhao et al. (2016).

This influence maximisation algorithm first simulates L-step walk R times for each node, then compute the influence value of each node according to the number of times each node being accessed, and finally applies the greedy method to select the top-k nodes.

3 Methodology

In this section, we first present a general sampling based framework to speed up the computation of centrality in large-scale graphs. To further show the effectiveness of this framework, we take the influence maximisation problem as an instance, and develop an algorithm called IM-RWS based on the sampling based framework. Finally, we introduce each step of the IM-RWS algorithm in details and analyse its time complexity.

3.1 The general sampling based framework

To speed up the computation of centrality in large-scale graphs, we propose a general sampling based method. The idea is as follows: We first do sampling on the original large graph and get multiple relatively small subgraphs, then compute the centrality values by analysing each of the sampled subgraphs, and finally aggregate the values to get the final result. Specifically, the method can be divided into the following four steps:

1 sample multiple subgraphs from the original large graph by using a certain sampling method

2 for each subgraph sampled in last step, calculate the centrality value with existing centrality computation algorithms

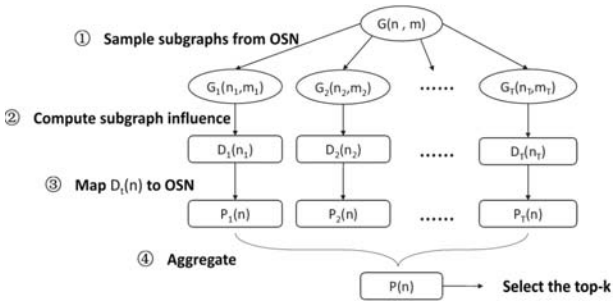
3 map the centrality values from the sampled subgraph to the whole large graph for each subgraph and then get the value distributions of the centrality values

4 aggregate the distributions to obtain the final solution.

Note that the above framework is general in the sense that it can be applied to speed up the computation of various graph centralities. In this paper, we take the influence maximisation problem as an example, and propose an IM-RWS algorithm to validate the feasibility of the above sampling based framework, and also use it to show the improvement it can achieve.

According to the sampling based framework, the IMRWS algorithm also consists of four steps. Specifically, we first sample a certain number of subgraphs from an OSN, then we calculate the distribution of the influence values of nodes in each subgraph. Next, we map this distribution to the original OSN. Finally, we make an aggregation of the distributions over each sampled subgraph to obtain the final influence value distribution of nodes in the entire OSN. Figure 1 shows the algorithm process of IM-RWS, and Algorithm 1 describes pseudocode. In the following subsections, we will introduce each step of the algorithm in details.

Figure 1 The process of IM-RWS algorithm



3.2 Sampling subgraphs from OSN

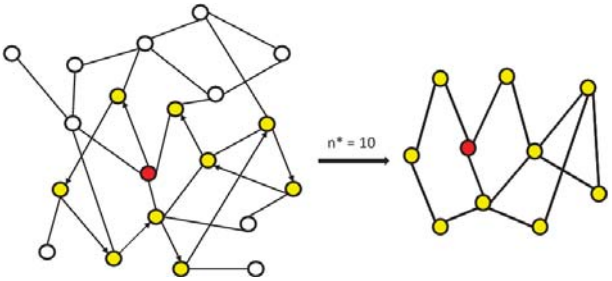
Firstly, we use random walk based sampling technology to obtain subgraphs. In particular, we run random walk on a given social network $G(V,E)$, and keep track of the visited nodes, and denote the set of nodes as V'. For any $u, v \in V'$, if the edge $uv \in E'$, then it will be remained in the subgraph. In other words, we take all visited nodes by the random walk, as well as all edges with both end nodes belonging to the visited node set as a sampled subgraph. For ease of presentation, we denote the set of the edges in the sampled subgraph as E', and denote the sampled subgraph as $G'(V', E')$. We repeat the sampling process T times to get T subgraphs. Figure 2 shows an example of this sampling process.

Algorithm 1 IM-RWS algorithm

Input: a graph $G(V, E)$ and a parameter k;
Output: a set S of k nodes for maximizing the influenced number of nodes $\sigma(S)$;
$S \leftarrow \emptyset$;
for $t = 0$ to T **do**
  $G_t \leftarrow \text{RWSampling}(G, n^*)$;
  /* Sampling subgraphs with random walks */
  $D_t(n_t) \leftarrow \text{computingInfluenceValue}(G_t)$;
  $P_t(n) \leftarrow \text{mappingToOriginalGraph}(D_t(n_t))$;
end for
$P(n) \leftarrow \text{Aggregating}(P_1(n), P_2(n), ..., P_T(n))$;
$\text{sort}(P(n))$;
for $i = 1$ to k **do**
  $v \leftarrow \text{index}(P(i))$;
  $S \leftarrow S \cup \{v\}$;
end for

In Figure 2, the left figure shows the original graph G, where the red node represents the start node, and the yellow nodes imply that they are visited by random walk, and the arrows show the path of the random walk. Suppose that the terminate condition is to sample 10 nodes, then the size of the subgraph is 10, which means $|V'| = 10$. After getting node set V', it is easy to get the edge set E'. The right figure shows the sampled subgraph G'. The corresponding algorithm is shown in Algorithm 2.

Figure 2 An example of random walk based sampling (see online version for colours)



Algorithm 2 RWSampling fuction

$G_t(V', E') \leftarrow \emptyset$;
while $|V'| < n^*$ **do**
  $v \leftarrow$ randomly pick a node in G;
  $V' \leftarrow V' \cup \{v\}$;
  while $|V'| < n^*$ and v's unvisited neighbor set $\neq \emptyset$ **do**
    $v \leftarrow$ randomly pick an unvisited neighbor of v;
    $V' \leftarrow V' \cup \{v\}$;
  end while
end while
for u in V' **do**
  for $\overrightarrow{v_i u} \in E$ **do**
    $E' \leftarrow E' \cup \{\overrightarrow{v_i u}\}$;
  end for
end for
return $G_t(V', E')$

3.3 Calculating influence values in subgraphs

For each subgraph obtained in the last step, we calculate the influence value of each node in each subgraph. Here, we follow the idea of the IM-RW algorithm (Zhao et al., 2016), which uses random walk to approximate the influence values. The main idea is to simulate R random walks of length L for each node, and according to the times of a node being visited by all random walks, we can determine its influence value. The detailed process is described as follows:

1 For $\forall i \in V'$, where V' represents the node set in the sampled subgraph. We simulate R times random walks of length L starting from i.

2 In each step of the random walk, we randomly pick one node from the neighbours of the current node, and walk to it in the next step.

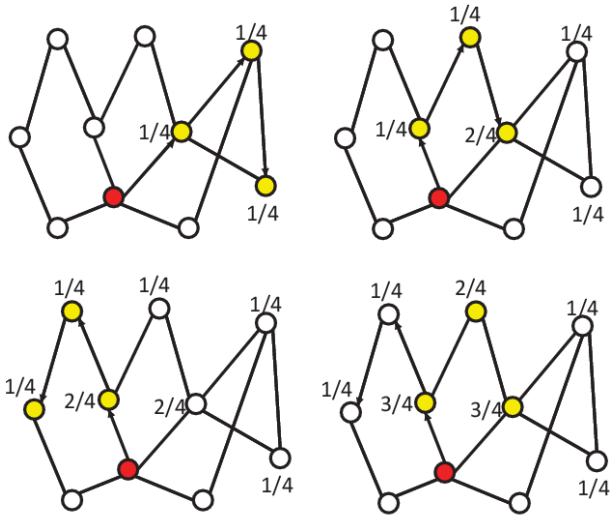
3 Repeat the last step until the length of the walk reaches L.

4 Initially we set $D(i) = 0$ for $\forall i \in V'$, where $D(i)$ presents the influence value of node i in the subgraph. Every time a node u is visited by a random walk, we plus $D(u)$ by $1/R$.

5 When R random walks are all performed, we can get a distribution of the influence values of all nodes in the subgraph, which is denoted as $D_t(n_t)$, where t denotes the id of current subgraph and $n_t = |V'|$.

Figure 3 shows an example to illustrate this process. The subgraph is obtained in the last subsection. In this subgraph, we simulate a 3-step random walk 4 times for each node. Initially, the influence values of all nodes are set as 0. When a node is accessed by a random walk, then its influence value increases by 1/4. After 4 random walks, we get the influence value distribution $D_t(n)$ of the current subgraph, as shown in Figure 3. The specific implementation is given in Algorithm 3.

Figure 3 Calculating influence values in subgraphs (see online version for colours)



Algorithm 3 computingInfluenceValue Fuction

for $i = 1$ to n^* **do**
  for $j = 1$ to R **do**
    $u \leftarrow G_t.v_i$;
    for $step = 1$ to L **do**
      if u's unvisited neighbor set $\neq \emptyset$ **then**
        $u \leftarrow$ randomly pick an unvisited neighbor of u;
        $D_t[u] \leftarrow D_t[u] + 1/R$;
      end if
    end for
  end for
end for
return D_t

3.4 Mapping $D_t(n)$ to OSN

Based on the obtained influence value distribution $D_t(n)$, in this subsection, we introduce how to map this distribution to the entire social network. In other words, for each subgraph, we will get a distribution over all nodes in the original graph. During this process, we define a function to reflect the influence value of each node as follows:

$$q_t(i) = \begin{cases} D_t(i), & \text{if } i \in G_t, \\ \sum_{j \in in-link[i]} p \cdot P_i(j), & \text{if } i \notin G_t, \end{cases} \quad (1)$$

where we define p as the decay factor. The physical meaning is that the influence will decrease as the distance increases. The process of the above mapping process is described as follows:

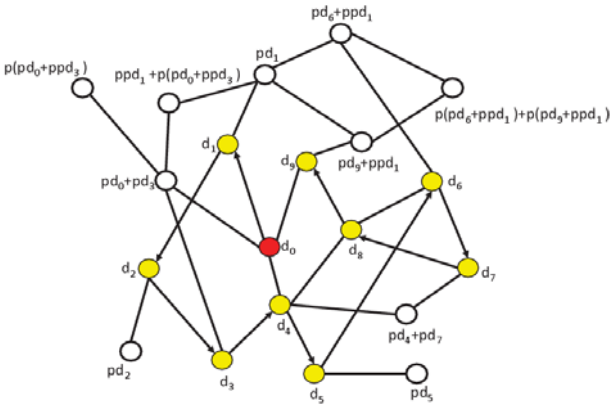
1 initially, we set $G(i) = 0$ for $\forall i \in V'$, where $G(i)$ presents the influence value of node i in the entire graph, and for the sampled nodes, we set $G(i) = D(i)$

2 define a queue Q, which is initialised as the sampled node set, to represent the node set which can spread the influence in the graph

3 in each iteration, we get a node u form the head of Q, then for each neighbour v of node u, we update $G(v)$ by setting $G(v) = G(v) + p * G(u)$, and push v into Q

4 repeat the process until the queue is empty.

Figure 4 An example showing the mapping of the influence from a subgraph to the original OSN (see online version for colours)



Algorithm 4 mappingToOriginalGraph Fuction

```
Define a queue Q;
for i = 1 to n do
    if i ∈ G_t then
        P_t[i] = D_t[i];
        Q.push(i);
    else
        P_t[i] = 0;
    end if
end for
while !Q.empty() do
    for i = 1 to d[i] do
        if step[link[i]] > step[i] then
            P_t[link[i]] += p * P_t[i];
            if link[i] has not been added in the queue then
                Q.push(link[i]);
            end if
        end if
    end for
end while
return P_t
```

Figure 4 shows an example of this process, where the red and the yellow nodes (d_0~d_9) belong to the current subgraph. The influence values are computed in above steps. For the nodes that are not in the subgraph, their influence values are computed by equation (1). The detailed implementation is given by Algorithm 4.

3.5 Aggregation

Since we have many subgraphs, which means each node will have many influence values after mapping. We need to aggregate them to get a final solution. Here we use a simple but efficient method by directly computing the average value and using the average value as the final influence value of each node. Finally, we sort the nodes by their influence values, and select the k nodes with the largest values as the most influential nodes.
The aggregation can be formulated as follows, and the detailed implementation is shown in Algorithm 5.

$$P(n) \triangleq \frac{1}{T}\sum_{t=1}^{T} P_t(n). \tag{2}$$

Algorithm 5 aggregation Fuction

```
for i = 1 to n do
    for t = 1 to n do
        P[i] += P_t[i];
    end for
    P[i] += P[i]/T;
end for
return P
```

3.6 Time complexity analysis

The time complexity of IM-RWS comes from four parts. In the first part, in order to get T subgraphs, we have to run random walk with T times, and each walk needs to sample n^* nodes, so the time cost in the sampling step is $O(Tn^*)$. In the second part, we compute the influence probability distribution $D_t(n^*)$ for T times since there are T subgraphs. In each computation, the time cost is $O(RLn^*)$, so the total time complexity in this part is $O(TRLn^*)$. In the third part, in order to map $D_t(n^*)$ to the entire OSN, we need to traverse the OSN, which leads to a time cost $O(n)$, where n is the total number of nodes in the OSN. Thus, the total time cost of this part is $O(Tn)$. The time cost in the fourth part is the same as that in the third part. So, from the analysis, we can conclude that the time complexity of our algorithm is $O(TRLn^*) + O(Tn)$. In experiment, we set $R = 100$, $L = 3$, $n^* = 1\%n$, and the time complexity is close to $O(Tn)$. We also find that $T = 10$ is already enough to achieve a good efficiency and accuracy.

4 Performance evaluation

In this section, we conduct experiments on several real-world datasets to compare our IMRWS algorithm with the state-of-the-art influence maximisation algorithm IMM and IM-RW from both aspects of accuracy and efficiency. Our algorithm is implemented in C++ and compiled with g++ 4.8.1. We run the experiments on a machine with an Intel Xeon E5-2407 v2 2.40 GHz*4 CPU and 48 GB memory.

4.1 Experimental settings

Datasets. The datasets we used in our experiments are all real-world graphs as shown in the Table 1. Among them, Yelp comes from Yelp Dataset (https://www.yelp.com/dataset_challenge/dataset), Flixster comes from Jamali and Ester (2010), Twitter comes from Twitter Dataset (https://an.kaist.ac.kr/traces/WWW2010.html) and all others come from Stanford Network Analysis Project (SNAP) (Leskovec and Krevl, 2014). We point out that the datasets include both small and large graphs, and in particular, the number of nodes of Twitter reaches several tens of millions, and the number of edges reaches one billion.

Table 1 Datasets

Dataset	*# of nodes*	*# of edges*
Ciao	2,342	57,544
NetHEPT	15,229	62,752
Epinions	18,089	355,813
Slashdot0811	77,360	905,468
Slashdot0902	82,168	948,464
Yelp	174,100	2,576,179
Flixster	300,000	6,394,798
LiveJournal	4,847,571	68,993,773
Twitter	41,652,230	1,468,365,182

Parameter settings. The parameters we used in our method are listed in Table 2, and the value settings of these parameters are listed in Table 3. In particular, in Table 3, SR is set as 1% and 0.01%, which are used in the first two sets of experiments and the last set of experiments, respectively.

Algorithms. In our experiment, we compare our IM-RWS algorithm with the state-of-the-art influence maximisation algorithm IMM and IM-RW in the aspects of accuracy and efficiency.

4.2 Results for efficiency

The efficiency of an algorithm is measured by the running time. Since the actual running time has a small fluctuation, we execute 10 times for each experiment and take the average value. Figure 5 shows the efficiency results for IMM, IM-RW and IM-RWS algorithms running on the eight datasets. The X-axis represents the seed set size *k*, varying from 5 to 50 with step 5, and the Y-axis represents the running time of each algorithm in seconds.

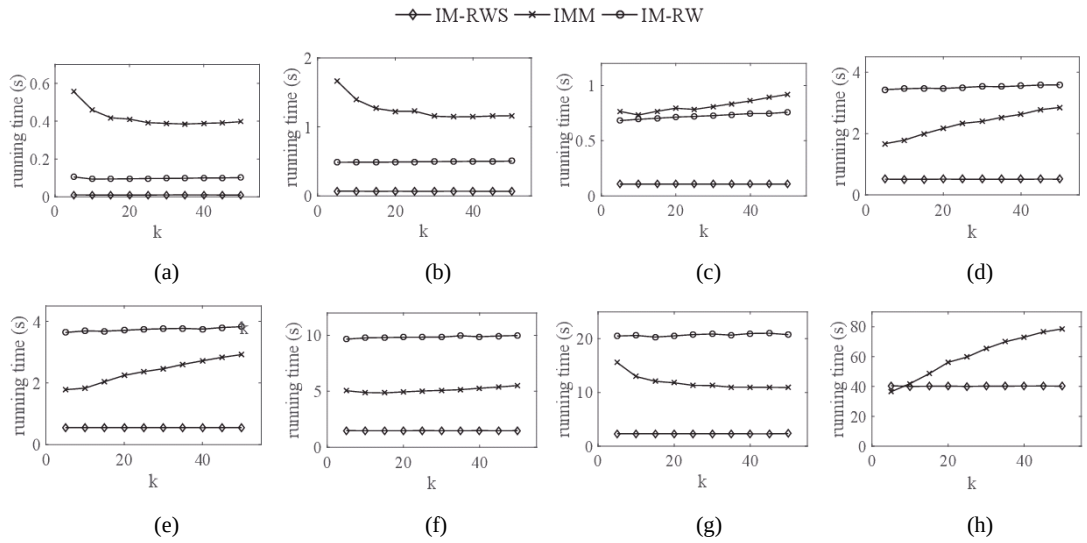
Table 2 Algorithm parameters

Parameter	*Description*
-G	Online social network G
-n	The number of nodes in G
-m	The number of edges in G
-d	Degree of every node in G
-k	The size of seed set
-T	The number of sampled subgraphs
-SR	Sample rate of each subgraph
-R	The number of random walks of each node
-L	The length of each random walk
-Model	Influence diffusion model

Table 3 Value setting of the parameters

Algorithm steps	*Parameters*	*Values*
Select top-k	-k	5 to 50 with step 5
Sample subgraphs from OSN	-T-SR	10 1%, 0.01%
Calculate influence	-R	100
value in subgraphs	-L	3
Compute influence	-Model	IC

Figure 5 Results of efficiency, (a) Ciao (b) NetHEPT (c) Epinions (d) Slashdot0811 (e) Slashdot0902 (f) Yelp (g) Flixster (h) LiveJournal



The results indicate that the running time has no significant variation with the increase of the seed set size *k*. By contract, for all eight graphs, our IM-RWS has a much shorter running time compared with IMM and IM-RW, with the efficiency increased by 4 to 8 times. Besides, for the largest graph LiveJournal [Figure 5(h)], IM-RWS costs only tens of seconds, while the running time of IM-RW is too long time so we do not show it in the figure to make the figure clear.

4.3 Results for accuracy

In our evaluations, we use the influence spread of the *k* starting nodes that are selected as the seed set to measure the accuracy of the algorithms. The more nodes the seed set can influence under the IC model in the whole graph *G*, the larger the influence spread is, so the higher accuracy the algorithm achieves.

Figure 6 shows the results of the accuracy for IMM, IM-RW and IM-RWS algorithms running on the eight datasets. The X-axis represents the seed set size *k*, varying from 5 to 50 with step 5. The Y-axis represents the number of nodes that can be influenced by the set of seed nodes under the IC model in the original graph *G*.

From the results we can see that with the increase of the seed set size *k*, the influence spread of IMM, IMRW and IM-RWS algorithms all increase in a nearly linear manner. Besides, IMM and IM-RW have the similar accuracy. From the comparison with our IMRWS algorithm, for the relatively small graphs Ciao [Figure 6(a)], NetHEPT [Figure 6(b)] and Epinions [Figure 6(c)], IM-RWS has a relatively lower accuracy, which is mainly caused by the sampling. However, with the increase of the graph size, the disparity is getting smaller. Finally, for the large graphs Slashdot0811 [Figure 6(d)], Slashdot0902 [Figure 6(e)], Yelp [Figure 6(f)] and Flixster [Figure 6(g)], our IM-RWS algorithm achieves even higher accuracy over IMM and IM-RW. To further show the effectiveness and efficiency of our algorithm on large graphs, we run another experiment by using a large dataset in the next subsection.

4.4 Results on large graph

For the Twitter graph we used, the number of nodes reaches tens of millions, and the number of edges reaches one billion. We believe that this graph is large enough to show the impact of our algorithm on large graphs. Also, this dataset is effective to further motivate our framework in speeding up the centrality computation as existing algorithm on this graph already takes a very long time. In this set of experiments, we compare our algorithm IM-RWS with IMM only, also from the aspects of accuracy and efficiency. Here, in IM-RWS, we set T as 10 and the sample rate as 0.01%.

Figure 7 shows the results on Twitter graph, where Figure 7(a) shows the accuracy and Figure 7(b) shows the results of efficiency. The X-axis represents the seed set size *k*, varying from 50 to 300 with step 50. The Y-axis represents the number of nodes that can be influenced by the seed set under the IC model in the whole graph *G* and the running time, respectively.

From the results we can see that IM-RWS can get a large improvement both in accuracy and efficiency compared with IMM. For accuracy, the influence spread of IM-RWS is almost twice as that of IMM as shown in Figure 7(a). For efficiency, as shown in Figure 7(b), IMM takes several minutes to run out. Moreover, as *k* increases, the running time also increases. However, IM-RWS only needs about 24 seconds, and more importantly, its running time is independent on *k*, which denotes the number of initial nodes being influenced. This implies that our algorithm IM-RWS has a very good scalability in both graph size and initial seed set size.

Figure 6 Results of accuracy, (a) Ciao (b) NetHEPT (c) Epinions (d) Slashdot0811 (e) Slashdot0902 (f) Yelp (g) Flixster (h) LiveJournal

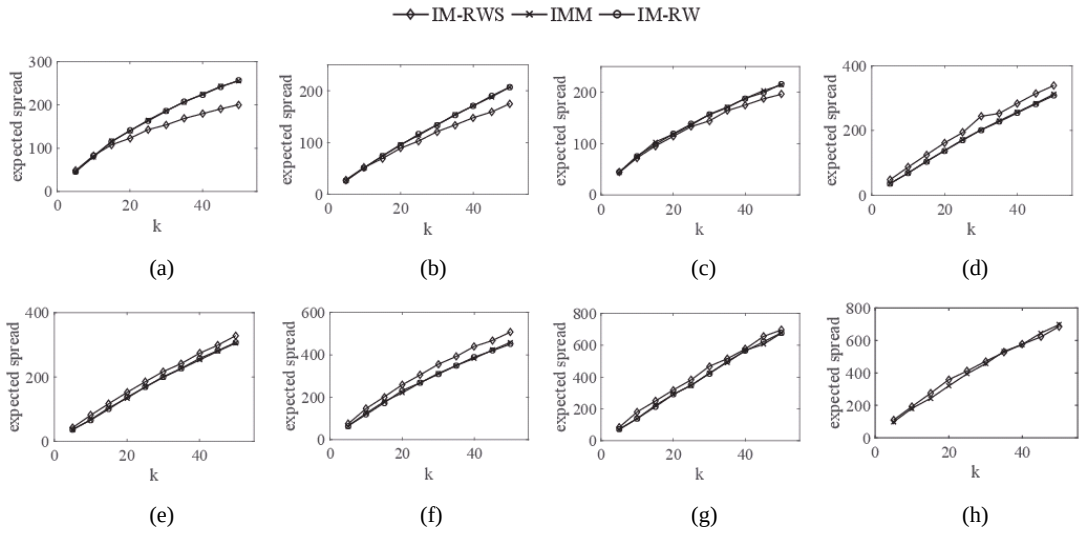
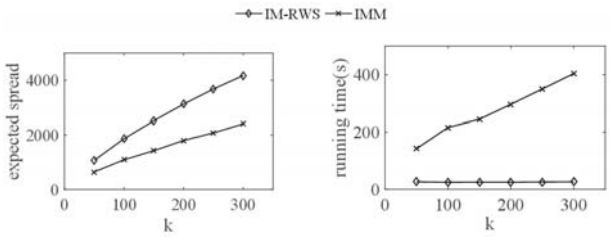


Figure 7 Results on a large graph Twitter, (a) accuracy (b) efficiency



(a) (b)

In summary, with the increase of graph size, our IM-RWS algorithm has higher accuracy. Moreover, the larger the graph is, the more accurate results we can achieve. This conforms to our original intension to design IM-RWS with the idea of sampling. The main reason is that for large-scale graphs, the traditional way of direct computation on the whole graph will be trapped in a relatively small area, and thus causes estimation error. With the increase of graph size, the error also increases. In a contrast, our sampling based IM-RWS algorithm disperses the computation over the graph and thus reduces estimation error. In terms of efficiency, with sampling, the time complexity of IM-RWS does not strictly depend on the size of the original graph, but mainly depend on the number of sampled nodes and the times of sampling. As a result, IM-RWS is also time-efficient to analyse large sized graphs.

5 Related work

As a seminal work, Kemple et al. (2003) make a great contribution on the influence maximisation problem. They first formulate the influence maximisation problem as a discrete optimisation problem and show that it is *NP-hard* in general. By taking use of the idea of approximation, they design a greedy algorithm which can give a guarantee on the result with a factor of $1 - 1/e$ under the triggering model. However, it leads to a high time cost of $O(knmr)$, which makes it unfeasible to be applied to a large-scale OSN. Motivated by this, lots of techniques are proposed to improve the efficiency. Among these works, CELF and CELF++ (Goya et al., 2011; Leskovec et al., 2007) are developed based on the submodularity of the spread function, and they also have a significant improvement on efficiency. However, due to the framework of the greedy algorithm, their time complexity still retains $O(knmr)$. In the meantime, many researchers design lots of heuristic algorithms, which can offer a higher efficiency, but none of them can give a theoretical guarantee on the accuracy of the results. Recently, Borgs et al. (2014) make a theoretical breakthrough on the influence maximisation problem, and they present an algorithm which can not only give a guarantee on accuracy, but can also reduce the time complexity to $O(kl_2(m + n)(logn)_2/\varepsilon_3)$. However, the small ε means that the time complexity still has a large constant factor, which incurs high overheads in practice. Motivated by Borg et al.'s idea. Tang et al. (2014, 2015) propose two algorithms to improve the efficiency, TIM and IMM, and IMM is now the state-of-art influence maximisation algorithm in terms of efficiency.

With respect to the sampling approaches, they can be roughly classified into four categories (Krishnamurthy et al., 2005; Leskovec and Faloutsos, 2006; Leskovec et al., 2005; Luo et al., 2008). The first category is random node sampling, which includes RN, RPN, RDN. The second category is random edge sampling, which includes RE, RNE, HYB, and so on. The third category is heuristics based sampling, which includes RNN, RW, and FF. The last category is the traditional graph traverse methods, which mainly include BFS, DFS, etc.

6 Conclusions

In this paper, we present a sampling-based method to compute graph centrality in large online social networks. In particular, to validate the effectiveness of the method, we take the influence maximisation problem as an application example, and present an IM-RWS algorithm to solve the influence maximisation problem by introducing the sampling idea to the existing influence maximisation algorithm IM-RW. The experiments show that our algorithm can achieve a significant improvement in efficiency, and it also improves the accuracy for middle and large OSNs, compared with the state-of-the-art influence maximisation algorithm IMM and IM-RW. For the future work, we will continue to focus on the influence maximisation problem, but try other methods in the second step for calculating the influence in subgraphs, so as to validate the generality of the methods of our sampling method and try to find a better algorithm for solving the influence maximisation problem. In addition, we also prefer to consider the implementation of our sampling based method to other graph centrality computation problems, such as similarity, so as to further validate the generality of our sampling based method.

References

Borgs, C., Brautbar, M., Chayes, J. and Lucier, B. (2014) 'Maximizing social influence in nearly optimal time', *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp.946–957.

Brandes, U. (2001) 'A faster algorithm for betweenness centrality', *Journal of Mathematical Sociology*, Vol. 25, No. 2, pp.163–177.

Cao, T., Wu, X., Wang, S. and Hu, X. (2010) 'OASNET: an optimal allocation approach to influence maximization in modular social networks', in *Proceedings of the 2010 ACM Symposium on Applied Computing*.

Chen, W., Yuan, Y. and Zhang, L. (2010) 'Scalable influence maximization in social networks under the linear threshold model', in *2010 IEEE International Conference on Data Mining*.

Domingos, P. and Richardson, M. (2001) 'Mining the network value of customers', in *ACM SIGKDD*.

Even-Dar, E. and Shapira, A. (2007) 'A note on maximizing the spread of influence in social networks', in *International Workshop on Web and Internet Economics*, Springer.

Galstyan, A., Musoyan, V. and Cohen, P. (2009) 'Maximizing influence propagation in networks with community structure', *Physical Review E*, Vol. 79, No. 5, pp.1–7, 056102.

Gao, F., He, J. and Ma, S. (2014) 'Modelling the relationship between trust and privacy in network environments', *International Journal of Computational Science and Engineering*, Vol. 9, No. 4, pp.347–354.

Goyal, A., Lu, W. and Lakshmanan, L.V. (2011) 'Celf++: optimizing the greedy algorithm for influence maximization in social networks', in *Proceedings of the 20th International Conference Companion on World Wide Web*, ACM.

Granovetter, M. (1978) 'Threshold models of collective behavior', *American Journal of Sociology*, Vol. 83, No. 6, pp.1420–1443.

Jamali, M. and Ester, M. (2010) 'A matrix factorization technique with trust propagation for recommendation in social networks', in *ACM RecSys*.

Jeh, G. and Widom, J. (2002) 'SimRank: a measure of structural-context similarity', in *ACM SIGKDD*.

Kempe, D., Kleinberg, J. and Tardos, E. (2003) 'Maximizing the spread of influence through a social network', in *ACM SIGKDD*.

Krishnamurthy, V., Faloutsos, M., Chrobak, M., Lao, L., Cui, J-H. and Percus, A.G. (2005) 'Reducing large internet topologies for faster simulations', in *International Conference on Research in Networking*, Springer.

Leskovec, J. and Faloutsos, C. (2006) 'Sampling from large graphs', in *ACM SIGKDD*.

Leskovec, J. and Krevl, A. (2014) 'SNAP datasets: Stanford large network dataset collection', June 2014 [online] http://snap.stanford.edu/data (accessed 1 May 2016).

Leskovec, J., Kleinberg, J. and Faloutsos, C. (2005) 'Graphs over time: densification laws, shrinking diameters and possible explanations', in *ACM SIGKDD*.

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J. and Glance, N. (2007) 'Cost-effective outbreak detection in networks', in *ACM SIGKDD*.

Li, R-H., Yu, J.X., Huang, X. and Cheng, H. (2014) 'Random-walk domination in large graphs', in *2014 IEEE 30th International Conference on Data Engineering*, IEEE.

Luo, Y., Joshi, A., Phansalkar, A., John, L. and Ghosh, J. (2008) 'Analysing and improving clustering based sampling for microprocessor simulation', *International Journal of High Performance Computing and Networking*, Vol. 5, No. 4, pp.200–214.

Mahajan, V., Muller, E. and Bass, F.M. (1990) 'New product diffusion models in marketing: a review and directions for research', *Journal of Marketing*, Vol. 54, No. 1, pp.1–26.

Newman, M.E. (2005) 'A measure of betweenness centrality based on random walks', *Social Networks*, Vol. 27, No. 1, pp.39–54.

Page, L., Brin, S., Motwani, R. and Winograd, T. (1999) *The PageRank Citation Ranking: Bringing Order to the Web*, Technical Report, Stanford InfoLab, November [online] http://ilpubs.stanford.edu:8090/422/ (accessed 1 March 2016).

Richardson, M. and Domingos, P. (2002) 'Mining knowledge-sharing sites for viral marketing', in *ACM SIGKDD*.

Saito, K., Kimura, M., Ohara, K. and Motoda, H. (2009) 'Learning continuous-time information diffusion model for social behavioral data analysis', in *Asian Conference on Machine Learning*, Springer.

Schlegel, R. and Wong, D.S. (2015) 'Private friends on a social networking site operated by an overly curious SNP', *International Journal of Computational Science and Engineering*, Vol. 10, No. 3, pp.281–292.

Tang, Y., Shi, Y. and Xiao, X. (2015) 'Influence maximization in near-linear time: a martingale approach', in *ACM SIGMOD*.

Tang, Y., Xiao, X. and Shi, Y. (2014) 'Influence maximization: Near-optimal time complexity meets practical efficiency', in *ACM SIGMOD*.

Twitter Dataset [online] https://an.kaist.ac.kr/traces//WWW2010.html. (accessed 1 October 2016).

Wang, Y., Cong, G., Song, G. and Xie, K. (2010) 'Community-based greedy algorithm for mining top-k influential nodes in mobile social networks', in *ACM SIGKDD*.

Watts, D.J. (2002) 'A simple model of global cascades on random networks', *Proceedings of the National Academy of Sciences*, Vol. 99, No. 9, pp.5766–5771.

Weng, M.M., Hung, J.C., Weng, J-D. and Shih, T.K. (2016) 'The recommendation mechanism for social learning environment', *International Journal of Computational Science and Engineering*, Vol. 13, No. 3, pp.246–257.

Yelp Dataset [online] https://www.yelp.com/dataset_challenge/dataset (accessed 1 May 2016).

Zhao, J., Lui, J., Towsley, D. and Guan, X. (2014) 'Measuring and maximizing group closeness centrality over disk-resident graphs', in *Proceedings of the 23rd International Conference on World Wide Web*, ACM.

Zhao, J., Lui, J.C.S., Towsley, D., Wang, P. and Guan, X. (2015) 'A tale of three graphs: sampling design on hybrid social-affiliation networks', in *2015 IEEE 31st International Conference on Data Engineering*, IEEE.

Zhao, P., Li, Y., Xie, H., Wu, Z., Xu, Y. and Lui, J.C.S. (2016) 'Measuring and maximizing influence via random walk in social activity networks', *DASFAA*, Forthcoming, arXiv preprint arXiv:1602.03966.

Zhong, M. and Shen, K. (2006) 'Random walk based node sampling in self-organizing networks', *ACM SIGOPS Operating Systems Review*, Vol. 40, No. 3, pp.49–55.